CANDY/COLLEEN I/O SUBSYSTEM

The Candy/Colleen I/O Subsystem is described in a series of manuals (or "books") all contained in a single cover. The manual hierarchy is shown below.

KEYBOARD/DISPLAY
OPERATION
MANUAL

DISK
UTILITY
USER'S
GUIDE

Console
view of
system

CENTRAL I/O
USER'S
DESCRIPTION

USER PROGRAM/
DISK HANDLER
INTERFACE

User program
view of
system

CIO/HANDLER
INTERFACE
GUIDE

DISK FILE
MANAGER
DESCRIPITION

SYSTEM
HANDLER
DESCRIPTIONS

CASSETTE
MANAGER
DESCRIPITION

I/O subsystem
implementation
details

HANDLER/SIO
INTERFACE
GUIDE

SERIAL I/O
BUS PROTOCOL
DESCRIPTION

Table of Contents

Appendices

KEYBOARD/DISPLAY OPERATION MANUAL

2. SCREEN CHARACTERISTICS

The screen can be operated in any of 23 configurations (11 modes, with or without split screen, plus the system device, mode 0). These configurations can be categorized as being all graphics, all text, or split screen, for the purpose of our discussion.

GRAPHICS MODES

Nine of the twelve modes are graphics modes, with characteristics as shown in Book III, page 18. Depending upon the mode, a 2-16 color selection is available for each pixel. Both left and right margins are movable, but the margins are meaningless in any mode but Mode 0.

The cursor is invisible in all modes but Mode 0 and there is no logical data structure between the entire screen and the pixel--that is to say, no logical lines or records.

NOTE: The changes made to numbers of modes reflect GTIA additions.

TEXT MODES

Three of the twelve modes are text modes, with characteristics as shown in Book III, page 18. The left and right margins are movable (under program control, but not as a built-in keyboard function). Margins have no effect in any but Mode 0.

The cursor is visible as the inverted video representation of the character on which it resides; it does not blink. The text screen is divided into logical elements called logical lines, each logical line being comprised of one to three physical lines (of data, not raster lines). The number of physical lines used to comprise a logical line is always the minimum required to hold the data for that line; logical lines on the screen need not all have the same length.

NOTE: This is only valid for Mode 0. Modes 1 and 2, although they print characters on the screen, act like graphics modes and should be treated as such.

If a user types more than three physical lines on the screen and hits "CR," only the first three physical lines are returned (followed by EOL). The remaining data define a new logical line.

When in Mode 0, and using the Screen Editor, there are many screen editing functions supported, as described in Section 3.

Actions which may change the number of physical lines comprising a logical line are:

1. Entering data when the cursor is at the right screen margin.

2. Inserting characters such that non-blank data is pushed past the right screen margin.

3. Deleting characters until an entirely blank physical line is produced.

4. Insert line, where the last logical line is pushed off the bottom of the screen.

SPLIT SCREEN CONFIGURATIONS

In split screen configurations, the bottom of the screen is reserved for four lines of text (40 characters per line, as in Mode 0). The text region is controlled by the screen editor device (E:) and the graphics region is controlled by the display device (S:).

Two cursors are maintained in this mode of operation, and the two screen portions follow the rules described earlier for Graphics Modes and Text Modes.

3. Keyboard Characteristics

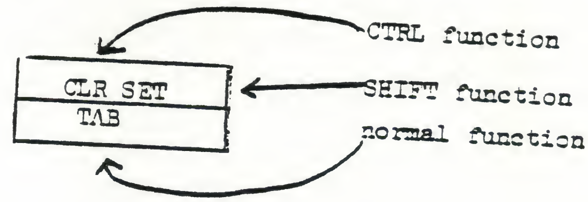
The keyboard is physically layed out as shown on the following page. Each keystroke causes a data byte to be made available to the system (which it may ignore if no program wants the data). Holding a key down for longer than one second will cause the corresponding key code to be made available approximately 10 times per second until the key is released.

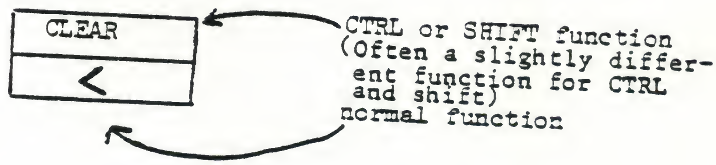
All keys produce unique codes except for the SHIFT key and the CTRL key; these two keys are always used in conjunction with one of the code producing keys, and act as modifiers. In the paragraphs that follow CTRL-x will indicate that key x is pressed while holding down the CTRL key and SHIFT-x will indicate a similar sequence but using the SHIFT key. Sequences using both the CTRL and SHIFT keys are invalid and will be ignored.

The key caps are marked so that their normal functions are indicated as well as their CTRL and SHIFT functions, in many cases.


CLEAR
INSERT
DELETE
BACK S
BREAK
CLR
RETURN
CTRL
CAPS
LOWR
SHIFT
MODEL No C7000

See the examples below.

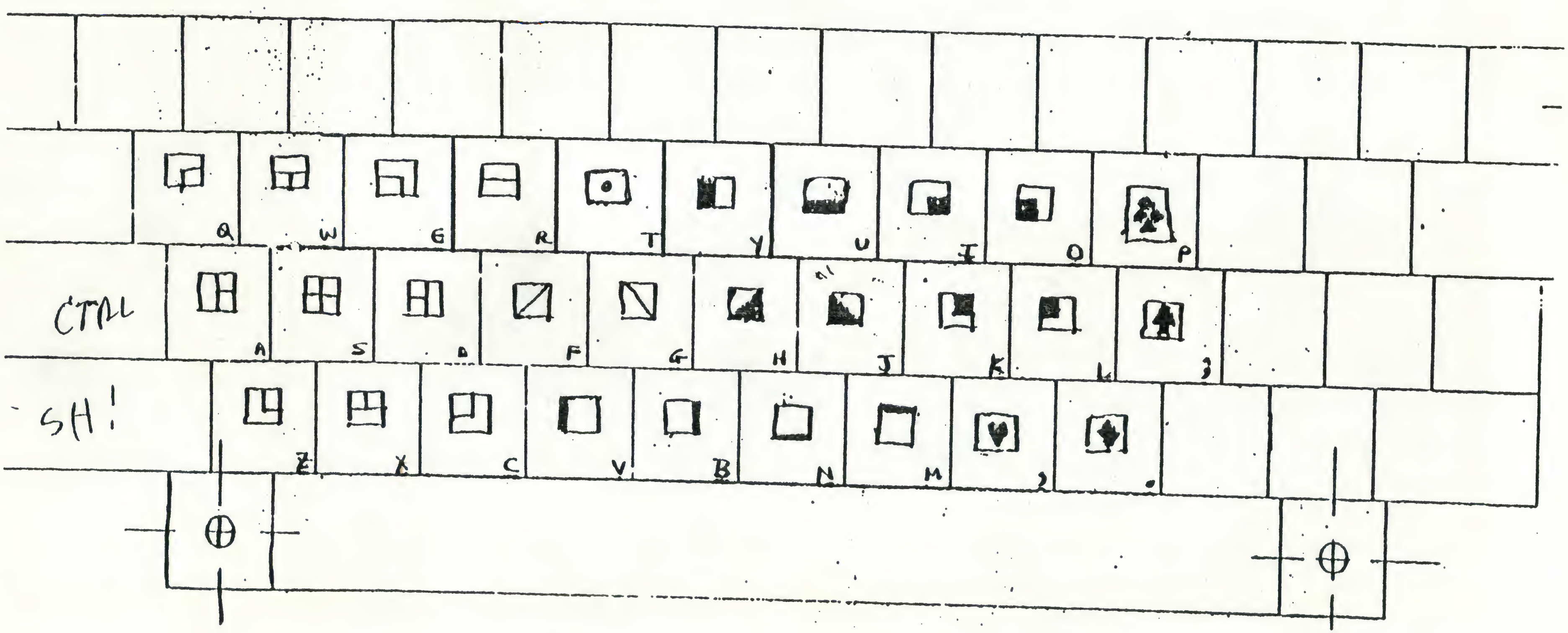




In addition to the functions shown, a set of special graphics characters are available as the CTRL functions of certain keys, as shown on the next page.

The internal codes for all of the valid key combinations and displayable characters are shown in Appendix L.

4-Dec-78



H. Bornstein 10-16-78

The following functions apply to all data read from the keyboard.

INVERT ( )

Toggles an internal switch that controls whether data being entered will be displayed in its normal or video inverse form.

Caps Lock (SHIFT-CAPS)

Alpha data entered will be in upper case form.
(This is the power on default.)

CTRL Lock (CTRL-CAPS)

Alpha data entered will be in character graphics from.

Unlock (CAPS)

Resets either of the lock functions, if set.

Shifted Data (SHIFT-data key)

Allows entry of the uppercase form of any data key. This key temporarily over-rides the Caps Lock and CTRL Lock settings.

Character Graphics Data (CTRL-data key)

Allows entry of the character graphics form of any data key. This key temporarily over-rides the Caps Lock and CTRL Lock settings.

Special Cases

CTRL-1 (Screen Editor Output Start/Stop)

Allows the user to stop and resume text output to the screen; produces no user accessible ATASCII code, as it is a local function.

CTRL-3 (Keyboard End-of-file)

Returns an EOL character as data, with an END-OF-FILE status.

KEYBOARD CODE CONVERSION RULES

Keys will be converted as they are input per the attached conversion tables; where the column heading "L.C." stands for lower case, "U.C." stands for upper case, and "CTRL" stands for control. The following rules (which are ordered by priority) determine which column to use.

1. If the SHIFT key and the CTRL key are both depressed, the keystroke will be ignored.

2. If the CTRL key is depressed, use the CTRL column.
3. If the SHIFT key is depressed, use the U.C. column.
4. If the CTRL key is locked, use the CTRL column for A-Z and the L.C. column for all else.
5. If the SHIFT key is locked, use the U.C. column for A-Z and the L.C. column for all else.
6. Use the L.C. column.

After the above stated conversion has been performed, and if the resultant character is <u>not</u> a system function code, invert bit -7 of the character if the "INVERT" flag is set (based on toggle of 川 key).

KEYCODE TO ATASCII CONVERSION

Key Code	Key Cap	L.C.	U.C.	CTRL	Key Code	Key Cap	L.C.	U.C.	CTRL
00	L	6C	4C	0C	20	,	2C	5B	00
01	J	6A	4A	0A	33_{10} 21	SPACE	20	20	20
02	;	3B	3A	7B	22	.	2E	5D	60
03					35_{10} 23	N	6E	4E	0E
04					24				
05	K	6B	4B	0B	25	M	6D	4D	0D
06	+	2B	5C	1E	26	/	2F	3F	
07	*	2A	5E	1F	27		*	*	*
08	O	2F	4F	0F	28	R	72	52	12
09					29				
0A	P	70	50	10	2A	E	65	45	05
0B	U	75	55	15	2B	Y	79	59	19
0C	RET	9B	9B	9B	2C	TAB	7F	9F	9E
0D	I	69	49	09	2D	T	74	54	14
0E	-	2D	5F	1C	2E	W	77	57	17
15_{10} 0F	=	3D	7C	1D	2F	Q	71	51	11
10	V	76	56	16	30	9	39	28	
11					31				
12	C	63	43	03	32	0	30	29	
13					33	7	37	27	
20_{10} 14					34	BACKS	7E	9C	FE
15	B	62	42	02	35	8	38	40	
16	X	78	58	18	36	<	3C	7D	7D
17	Z	7A	5A	1A	37	>	3E	9D	FF
18	4	34	24		38	F	66	46	06
25_{10} 19					39	H	68	48	08
1A	3	33	23	*	3A	D	64	44	04
1B	6	36	26		3B				
1C	ESC	1B	1B	1B	3C	CAPS	*	*	*
1D	5	35	25		3D	G	67	47	07
30_{10} 1E	2	32	22	FD	3E	S	73	53	13
1F	1	31	21	*	3F	A	61	41	01

* = special handling

4. SCREEN EDITING

The keyboard and display may be logically combined for a mode of operation known as screen editing. While in this mode, many of the keys take on added function as described in the following paragraphs.

SCREEN EDITING FUNCTIONS

Clear Screen (CTRL-CLEAR, SHIFT-CLEAR)

Clears all data from screen and places cursor at the home position.

Insert Line (SHIFT-INSERT)

Moves the physical line in which the cursor resides, and all physical lines below that line, down one physical line. Note that the last logical line on the display may be truncated as a result. The new physical line becomes the beginning of a new logical line.

Insert Character (CTRL-INSERT)

Moves all data within a logical line, between the cursor and the end of the line, one position to the right, by inserting a space; the last character of the logical line will be lost when the logical line is full and an Insert Character is performed.

Delete Line (SHIFT-DELETE)

Removes the logical line in which the cursor resides and moves all data below that line upward to fill the void.

Delete Character (CTRL- DELETE)

Removes the character on which the cursor resides and moves all data to the right of the cursor (within the logical line) one position to the left.

Cursor Up (CTRL- ↑)

Moves the cursor up by one physical line; the cursor will wrap from the top line of the display to the bottom line.

Cursor Down (CTRL-↓)

Moves the cursor down by one physical line; the cursor will wrap from the bottom line of the display to the top line.

Cursor Right (CTRL-→)

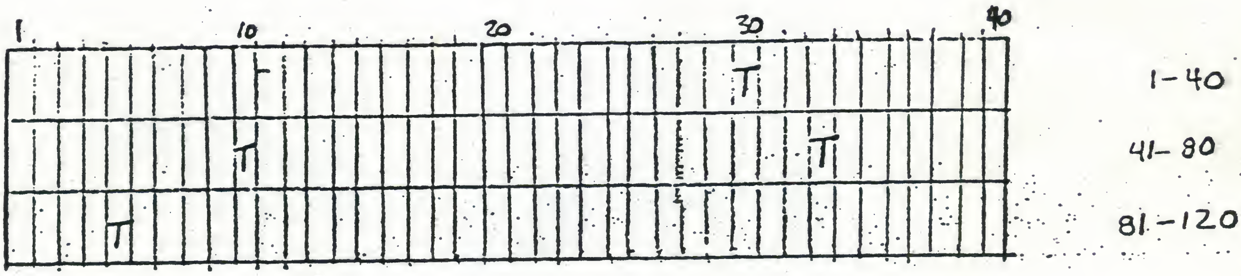
Moves the cursor right by one column; the cursor will wrap from the right margin of the display to the left margin (staying within the same physical line).

Cursor Left (CTRL-←)

Moves the cursor left by one column; the cursor will wrap from the left margin of the display to the right margin (staying within the same physical line).

Set Tab (SHIFT- TAB)

Establishes the logical line position at which the cursor is residing as a tab point. The logical line position is different than the column position, as a logical line may be up to three physical lines in length. For example, tabs may be set at positions 30, 50, 73 and 85 as shown below for a mode 0 logical line.



Clear Tab (CTRL-TAB)

Clears the logical line position at which the cursor is residing as being a tab point.

Tab (TAB)

Moves the cursor to the next tab point in the current logical line, or to the next logical line if no tab point is found. Note that this function will not increase the line length to accommodate a tab point outside the current length (e.g. logical line length is 40 characters and there is a tab point at position 60).

Backspace (BACK S)

Moves the cursor to the left one space (but not past the beginning of the logical line) and blanks the character at that position.

Escape (ESC)

Allows the keystroke that follows to be entered as data, even if it would normally be interpreted as a control function.

Carriage Return (RETURN)

Allows the logical line within which the cursor resides to be sent to the program which is in control of the display. The entire line will be sent (with the exception of trailing blanks), no matter where the cursor is positioned within the line. After all of the data in the logical line has been read, the cursor will be positioned to the beginning of the next logical line.

A special case occurs when characters are output without a terminating EOL, and then additional characters are appended to that logical line from the keyboard. When the RETURN key is pressed, only the keyboard entered characters will be sent to the program, unless the cursor has been moved out of and then back into the logical line, in which case, all of the characters will be sent.

If the logical line sent uses the last physical line of the display, all of the screen data will be scrolled upward by one logical line.

CENTRAL INPUT/OUTPUT SUBSYSTEM USER'S DESC.

1. INTRODUCTION

The Central Input/Output subsystem (CIO) provides the user programmer with a single interface to access all of the system peripheral devices, in a (largely) device independent manner. In addition, the following high-level features are available:

* Record or character aligned access
* Automatic blocking and de-blocking
* Concurrent access to multiple devices and files (non-overlapped)
* Error detection and recovery built in

2. CIO FEATURES

This section describes the basic capabilities of the CIO - those that are device independent; the device specific features are discussed in section 5.

2.1 DEVICE INDEPENDENCE

CIO provides device independence by having a single entry point and a device independent calling sequence; once a device or file is OPENED, data transfer can occur with no regard to the actual device involved. Uniform rules for handling character and record transfers make the device block sizes transparent to the user.

Status codes provided to the user are high-level numeric codes as described in Appendix C.

2.2 RECORD ACCESS METHODS

Two record access modes are provided by CIO-character aligned and record aligned.

Character aligned accesses allow the user to treat the device or file as a character stream; any number of characters may be read or written and the following operation will continue where the prior one left off. Records are of no consideration in this mode and reads or writes may encompass multiple records if desired.

Record aligned accesses allow the user to deal with the data stream at a higher level, that of the data record. Each and every write operation creates a record (by definition) and each read operation assures that the following read will start at the beginning of the next record. Record aligned accesses may not deal with portions of more than one record at a time.

2.3 MULTIPLE DEVICE/FILE CONCURRENCY

Up to eight devices and files may be accessed concurrently using CIO, each operating completely independently of the others.

2.4 UNIFIED ERROR HANDLING

All error detection and error recovery occurs within the CIO sub-system and status information that reaches the user is in the form of a standard error number.

3. CIO COMMAND OVERVIEW

This section describes the seven basic I/O commands that are supported by all of the system devices. Other, device specific, commands are described in section 5.

3.1 OPEN

Before a device may be accessed, it must be OPENED; this process links a specific I/O Control Block (ICOB) to the appropriate device handler, initializes the device or file, initializes any CIO related control variables and passes any device specific options to the device handler.

3.2 PUT RECORD

The user provides a data buffer address and the buffer length; CIO then transfers data from the buffer to the device handler until one of the conditions shown below occurs, at which time the indicated action is taken.

PUT RECORD

CONDITION	ACTION
1. EOL in buffer.	1. Transfer EOL; return with normal status.
2. No EOL in buffer.	2. Transfer EOL after last data byte.

3.3 PUT CHARACTER(S)

The user provides a data buffer address and the buffer length; CIO then transfers the entire contents of the data buffer to the device handler.

For the special case of PUTting one byte, the data may be passed in the A register instead of the user buffer if the buffer length bytes are set to zero.

3.4 GET RECORD

The user provides a data buffer address and the buffer length; CIO then transfers data from the device handler to the buffer until one of the conditions shown below occurs, at which time the indicated action is taken.

CONDITION	ACTION
1. EOL read and buffer not overflow.	1. Transfer EOL; return with normal status.
2. No EOL read and buffer full.	2. Read to EOL (throwing away data); return with "truncated record" status.

3.5 GET CHARACTER(S)

The user provides a data buffer address and the buffer length; CIO fills the buffer with data from the device handler.

For the special case of GETting one byte, the data is passed in the A register instead of the user buffer--if the buffer length bytes are set to zero.

3.6 CLOSE

To terminate all activity to a device or file, it must be closed. This process frees the associated IOCB, completes any pending data, PUTs and goes to the device handler for device specific actions.

3.7 GET STATUS

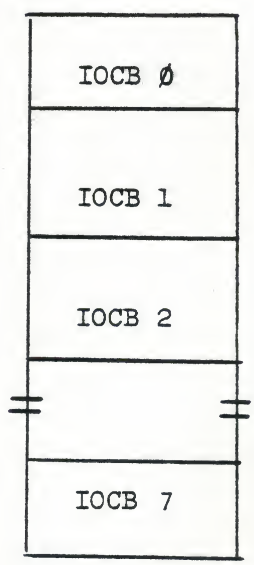
CIO calls the appropriate device handler which puts four bytes of device dependent status into system locations 'DVSTAT' through 'DVSTAT'+3.

NOTE: See Appendices H & I for the status byte formats.

4. CIO COMMAND INVOCATION PROCESS

The central element in performing I/O using CIO is the Input/Output Control Block (IOCB); all communication between CIO and the user regarding a specific operation is conveyed via this RAM resident structure.

There are eight IOCB's in the system which are arranged linearly in a table as shown below:



One IOCB is required for each currently OPEN device or file. Each IOCB may be used to control any of the system devices, although certain default assignments are made when the system is first powered up; namely, IOCB 0 is assigned to the screen editor (E:).

The IOCB format with the system names is shown in Appendix A.

A typical I/O operation is performed by having the user insert appropriate parameters into an IOCB, put the offset into the X register, and then JSR to the CIO routine. Upon return, the IOCB status byte may be interrogated to determine the degree of success of the requested operation. Status will also be in register Y.

see page III-38

The individual parameters of the IOCB are described in the following paragraphs.

4.1 HANDLER I.D.

The Handler I.D. is an index into the System Handler Table; This is provided by CIO as a result of an OPEN command and must not be altered by the user. This byte will equal FF_{16} for an unused or closed IOCB.

4.2 DEVICE NUMBER

The Device Number is provided by CIO as a result of an OPEN command and must not be altered by the user.

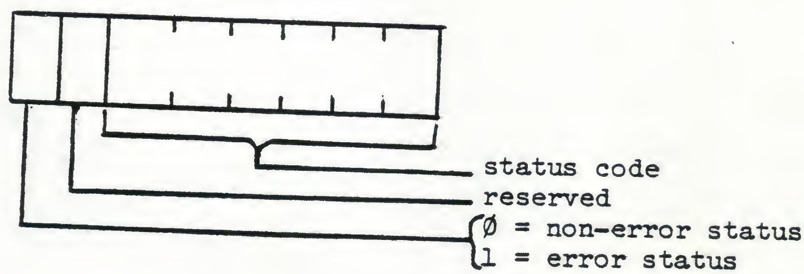
4.3 COMMAND BYTE

The Command Byte is provided by the user and is used to direct CIO and the device handler.

The command byte value for each of the commands will be found in Appendix B.

4.4 STATUS

The status byte is used by CIO to convey operation status to the user; see Appendix C for a list of the status codes. The general form of the status byte is shown below.



4.5 BUFFER ADDRESS

This two byte pointer is set by the user and is not altered by CIO. The user buffer has different uses for different commands as shown below.

OPEN - contains device/filename specification

GETXXX - receives data from device

PUTXXX - provides data for device

GETSTATUS - contains device/filename specification when device/ file not already OPEN.

4.6 PUT ADDRESS

This two-byte address is set to address-1 of the PUT byte routine of the device handler. This is provided by CIO as a result of an OPEN command and must not be altered by the user.

4.7 BUFFER LENGTH/BYTE COUNT

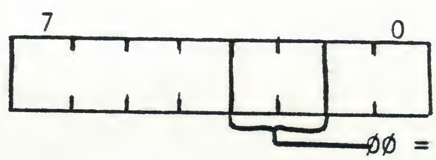
This two byte count is set by the user to indicate the size of the data buffer; no more than the indicated number of the bytes will ever be transferred into or out of the buffer by CIO. Upon return to the user, CIO will set this parameter to the number of bytes actually transferred.

The return byte count may be less than the buffer length due to: 1) errors, 2) end-of-file reached, 3) GET RECORD or 4) PUT RECORD.

4.8 AUXILLIARY INFORMATION

These two bytes are set by the user and contain information which is device dependent; these bytes are used during OPEN and in some of the device dependent commands to be explained in section 5.

For OPEN, two bits of AUX1 are always used to specify the direction of I/O as shown below.



ØØ = no I/O allowed
Ø1 = input only
1Ø = output only
11 = input/output

AUX1 should not be altered while a device is OPEN.

5. DEVICE SPECIFIC INFORMATION

This section describes, for each device, the CIO commands that are supported and details any device specific characteristics.

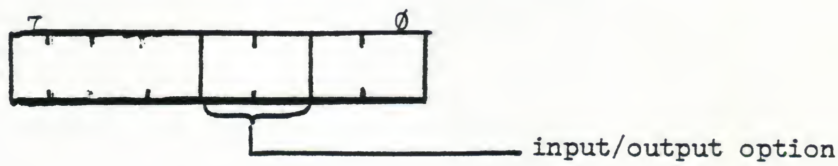
5.1 SCREEN EDITOR

This device uses the keyboard and display to simulate a screen editing terminal. Writing to this device causes data to appear on the data display starting at the current cursor position. Reading from this device activates the screen editing process and allows the operator to do data entry and editing. Whenever the RETURN key is pressed, the entire logical line within which the cursor resides is selected as the current record to be transferred by CIO to the user program. Trailing blanks are not transferred, however. When there is no active read to this device, screen editing may not be performed.

See I-15 and pages following for more information.

The symbolic device name of the screen editor is E and there is only one of them, so the device specification for OPEN is "E:

AUX1, in the IOCB, is interpreted as shown below at OPEN time.



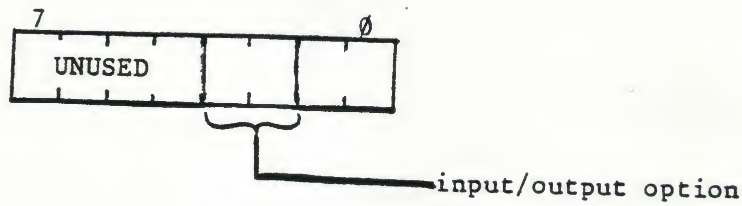
The IOCB record length parameter may be any value, but the screen editor will always start a record at the beginning of a display line. Records will not be "packed" on the display unless they are multiples of 40 bytes in length.

5.2 Keyboard Device

This device allows the user to read the converted (ATASCII) keyboard data as each key is pressed. Key data are not buffered, so any key strokes, performed while there is not an active read to this device, will be lost.

The symbolic device name of the keyboard is "K". There is only one of them, so the device specification for OPEN is "K:".

AUXI, in the IOCB, is interpreted as shown below at OPEN time:



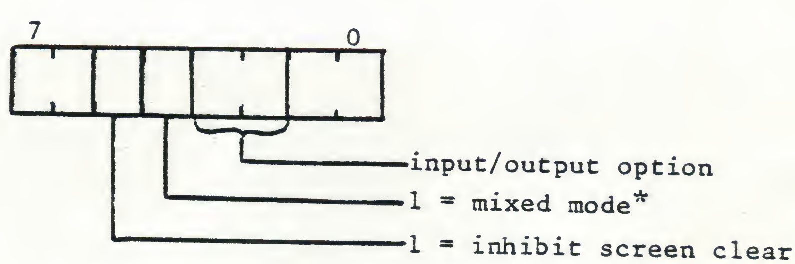
Note that the GET RECORD command does not provide editing of the data presented to the CIO caller; the caller gets every key stroke including editing keys and does not get any of the prior screen data.

5.3 DISPLAY DEVICE

This device allows the user to read characters from, and write character into, the display using the cursor as the screen addressing mechanism. Both text and graphics operations are supported.

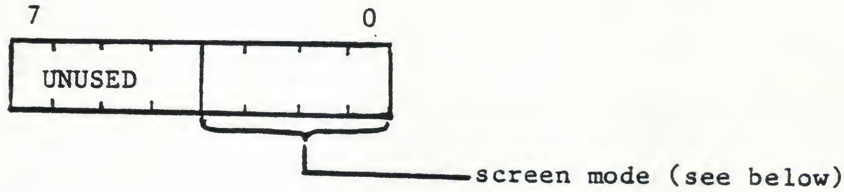
The symbolic device name of the display (screen) is "S" and there is only one of them, so the specification for OPEN is "S:".

AUXI, in the IOCB is interpreted as shown below at OPEN time:



*Due to hardware constraints, mixed mode is not allowed for Modes 9, 10, and 11.

AUX2, in the IOCB, is interpreted as shown below at OPEN time:



The screen modes and their characteristics are shown below:

Mode #	Type	ADDRESSABLE POINTS Horiz.	Vert.	Colors	Character Set	Displ. Memory	Displ. List	Split Screen Reg.
0	characters	40	24	2	All	960		
1	"	20	24	5	Upper Case & numbers OR Lower Case & Char.Grph.	480		
2	"	20	12	5		240		
3	Graphics	40	24	4		240		
4	"	80	48	2		480		
5	"	80	48	4		960		
6	"	160	96	2		1920		
7	"	160	96	4		3840		
8	"	320	192	2		7680		
9	"	80	192	16 lum.		7680		
10	"	80	192	9 col/ lum. reg.		7680		
11	"	80	192	16 color		7680		

GTIA

The mixed mode option modifies the screen mode selected to include four text lines of Mode #0 at the bottom of the display screen. In order to use mixed mode, both the screen editor (E:) and the display (S:) must be OPEN, and the display must be OPENed after the screen editor.

The inhibit screen clear option allows the user to OPEN the display without clearing the display memory. Mode changes may make the display change even though the display memory is unchanged, however.

Whenever display data are sent to the display, the cursor is moved after the data are put to the screen. If data are put to the lower right corner of the screen and the cursor is not moved (as described on the next page), then the next data character to be sent will produce an I/O error and the cursor will be sent to the home position (even if the character is CLEAR SCREEN).

Certain functions of the keyboard/display handlers require the user to examine and/or alter system memory; the paragraphs that follow, give the standard names and brief descriptions of the use of the variables involved.

COLOR REGISTERS

COLOR Ø

COLOR 1

COLOR 2

COLOR 3

COLOR 4

These variables contain color/lum values for the corresponding hardware registers. The RAM data is sent to the hardware as part of the system VBLANK interrupt processing.

CURSOR

ROWCRS

COLCRS double byte: LO, HI

These variables may be examined to determine the current screen cursor position or may be altered to force the cursor to a new position. When the cursor values are changed, the cursor will not move until the next I/O operation involving the display.

The second byte (HI bits) of COLCRS will always be zero except when the screen is in mode 8.

When the screen is in a split screen configuration, a separate pair of cursor registers are used to control the text regions cursor at the bottom of the screen.

TXTROW

TXTCOL double byte: LO, HI

The upper left corner of each of the two split regions is defined as 0,0 for the region's associated cursor.

CURSOR INHIBIT

CRSINH

When this flag is set non-zero, the text cursor will not display.

DISPLAY FLAG

DSPFLG

This variable is used to specify whether control characters sent to the display are to be displayed as special graphics or processed normally (and not displayed).

DSPFLG $\neq \emptyset$, display and don't process

$= \emptyset$, process and don't display

CHARACTER SET SELECT

CHBAS

This variable selects either uppercase and numbers or lower case and character graphics as the character set when in one of 20 character text modes (1 or 2).

CHBAS = $E\emptyset_{16}$ for upper case and numbers

= $E2_{16}$ for lower case and character graphics

NOTE: CHBAS is restored to $E\emptyset_{16}$ when any screen or editor mode is OPENed.

KEYBOARD DATA

CH

Key data that has been entered from the keyboard is stored in CH (by the interrupt service routine); when the data is later seen by the keyboard handler, CH is set to FF_{16} (an invalid key code).

If the user wants to monitor for keyboard activity, but not to give up control until the next keystroke, he monitors CH until it is not equal to FF and then initiates a GET CHARACTER I/O call from the K:device to get the ATASCII data. Note that CH contains the keyboard matrix code, not the ATASCII code for the key stroke.

MARGIN CONTROL

LMARGN

RMARGN

LMARGN establishes the left margin of the screen for Mode Ø only. The minimum value is zero and the system default value is two. RMARGN establishes the right margin of the screen for mode Ø only. The maximum meaningful value is 39; if the value is larger than that, it will be treated is if it were 39 anyway. The system default value of RMARGN is 38.

When in effect, the margins define the accessable portion of the display; the cursor (and hence the data) is limited to the inclusive bounds defined by the margins.

Data going to or from the display, when in one of the graphics modes, is in one of the formats shown below:

7							Ø	
X	X	X	X	X	X	X	D	Modes 4, 6, 8
X	X	X	X	X	X	D	D	Modes 3, 5, 7
X	X	X	X	D	D	D	D	Modes 9, 10, 11

D = "DATA"

X = "DON'T CARE"

The DATA portion (D) contains the bit(s) corresponding to a single pixel. Two exceptions to the DON'T CARE bits are CR($9B_{16}$) and CLEAR SCREEN ($7D_{16}$). If their code is sent to either E: or S:, they will be performed. Any data from the display, however, will have all DON'T CARE bits zeroed.

As graphics data are read/written, the cursor moves to the position after the last accessed pixel (as in text mode). Thus, a PUT of multiple characters will draw a horizontal line segment or a horizontal band.

In addition to plotting data point by point, two special commands are available to perform higher level functions: DRAW and FILL.

DRAW

This function draws a "straight" line from the actual cursor location to the location specified in ROWCRS and COLCRS. The IOCB command byte = 11_{16} and other user settable portions of the IOCB are not used. The color of the line drawn is the same as the last pixel updated prior to the DRAW command.

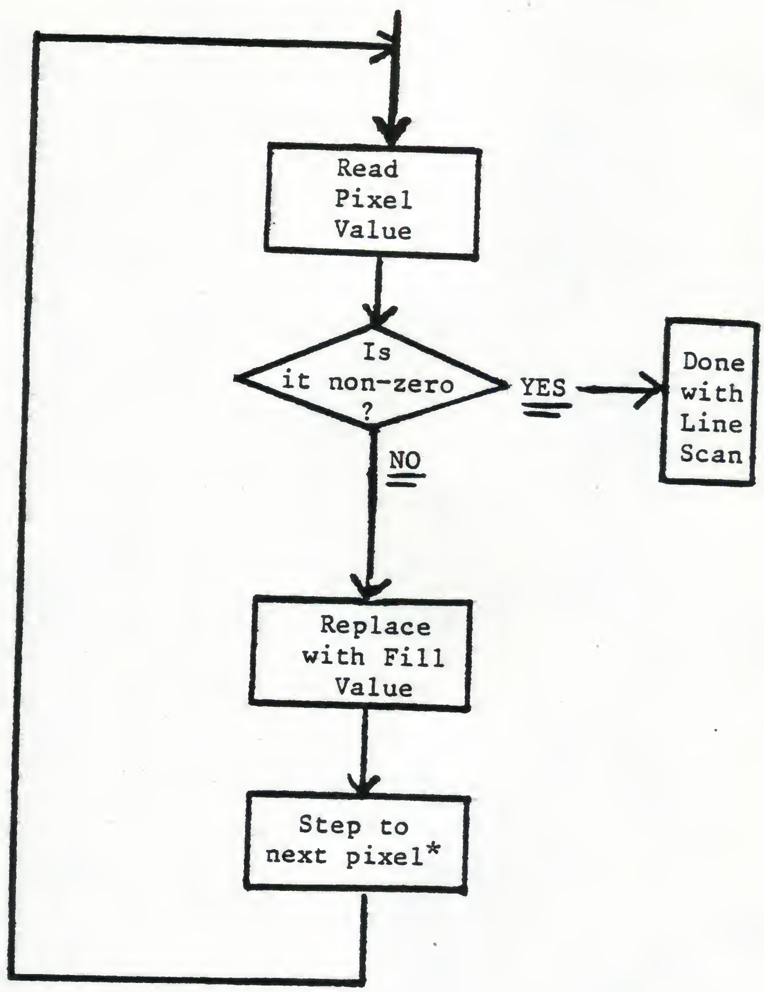
At the completion of this command the actual cursor will be at the location specified in ROWCRS and COLCRS.

FILL

FILL is a special function used to fill in areas of back ground with a specified color. The command is setup similar to DRAW, but as each point of the line is drawn, the rountine scans to the right performing the procedure shown on the next page.

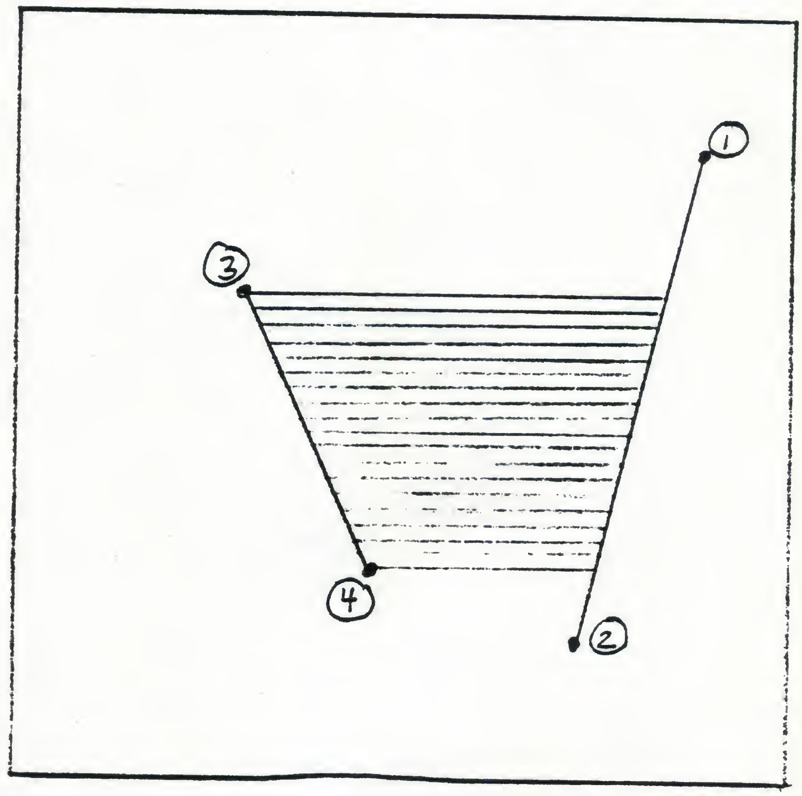
FILDAT contains the fill data and ROWCRS and COLCRS contain the cursor coordinates of the line endpoint. The IOCB command byte = 12_{16} and other user settable portions of the IOCB are not used.

FILL Line Scan



*End of line wraps to beginning of same line.

FILL example



1 - Set cursor and plot point

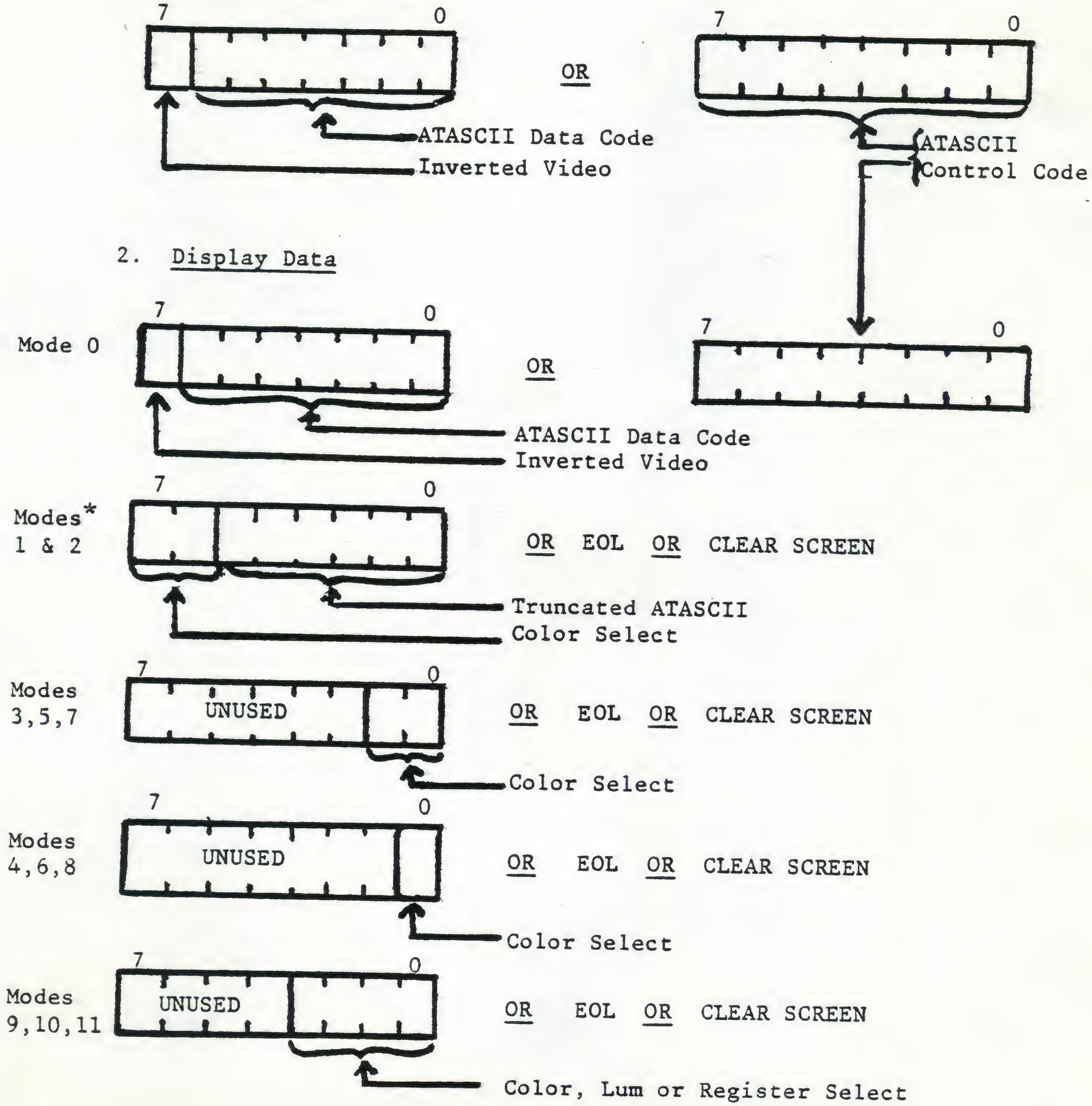
2 - Set cursor and DRAW line

3 - Set cursor and plot point

4 - Set fill value, set cursor, and FILL.

KEYBOARD/DISPLAY DATA FORMATS

1. Keyboard Data (mode independent)



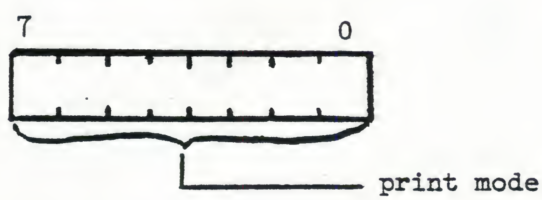
*NOTE: A "[" character + color select = 10 is indistinguishable from an ATASCII EOL.

5.4 Printer

This device prints ATASCII characters, a line at a time; it recognizes no control characters and neither does it's handler (except for EOL). See Appendix I for the printer character set.

The symbolic device name of the printer is P. The specification for OPEN is "P:".

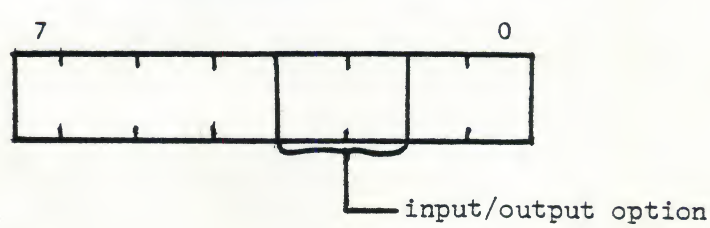
AUX2, in the IOCB, is used in conjunction with the write commands to specify one of three modes as shown below:



00 OR $4E_{16}$('N') ≡ normal (40 upright characters)

53_{16}('S') ≡ sideways (16 sideways characters)

AUX1, in the IOCB, is interpreted as shown below at OPEN time.

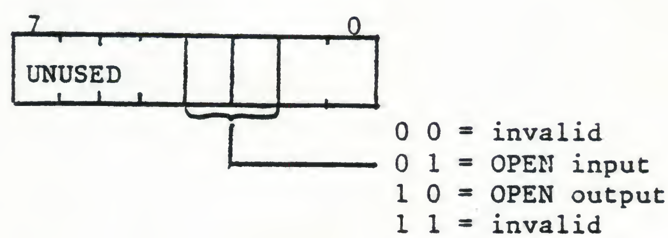


5.5 Cassette Device

The cassette is a read/write device OPEN either for reading or writing but never both, as all output files start at (or define) the logical end of tape.

The symbolic device for the cassette is C and there is only one of them, so the device specification for OPEN is "C:".

AUXI, in the IOCB, is interpreted as shown below during the OPEN operation.



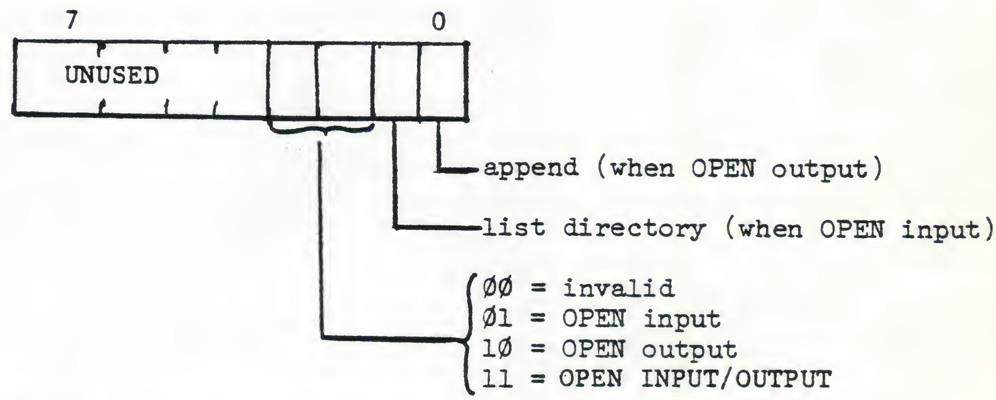
AUX2: 00 normal IRG mode
      80 short IRG mode

In short IRG mode, the tape does not stop throughout the reading/writing of the entire file; on reading, if read requests are not made within a certain time frame, data will be missed.

5.6 DISK File Manager (FMS)

The symbolic device name for the disk is D and there may be up to 4 of them, so the device specification for OPEN is "D [<n>] : <filename>EOL" where <filename> is a string of up to 12 characters. If <n> is not specified, it defaults to the value 1.

AUX1, in the IOCB, is interpreted as shown below during the OPEN operation.



The "append" option is used to add data to the end of an existing file.

The "list directory" option allows the caller to read directory information using normal GET RECORD or GET CHARACTER commands: the information will be in Atari ASCII suitable for outputting to any other device (i.e. formatted for display or print).

The Disk File Manager provides several commands which are unique to this handler and reflects the multiple file organization of the disk. These commands are:

RENAME	file(s)
DELETE	file(s)
LOCK	file(s)
UNLOCK	file(s)

<u>RENAME</u> - allows the user to change the name of any of a number of named files; the FMS expects to find a device specification plus a second name in the user buffer, as shown below:

D [<m>]:<filename> , <filename>

All occurrences of the first filename will be replaced with the second filename (using the wildcard rules as explained on page III-[redacted]35. No protection is provided against forming duplicate names, and once formed, duplicate names cannot be separately RENAMED or DELETED; however, an OPEN command will always select the first one found so that one of the duplicates is always accessible. The RENAME command does not alter the content of the files involved, merely the name in the directory.

DELETE - allows the user to delete any number of named files from the disk directory and deallocate the disk space used by the files involved. The FMS expects to find a device specification in the user buffer; all occurrences of the filename will be deleted (using the wildcard rules as explained on page III-35).

LOCK - allows the user to limit access to any number of named files. The FMS expects to find a device specification in the user buffer. LOCKED files may not be DELETED, RENAMED, nor OPENED for output until UNLOCKED. LOCKing a file that already is LOCKed is a valid operation.

UNLOCK - allows the user to remove the LOCK status of any number of named files. The FMS expects to find a device specification in the user buffer. UNLOCKing a file that is not LOCKed is a valid operation.

The Disk File Manager also provides two more commands that are used for random access to the disk:

NOTE disk address

POINT to disk address

NOTE - returns to the caller two values that together give the exact location of the next byte to be read or written. The absolute disk sector number is returned to the IOCB in ICBLL (LSB) and ICBLH (MSB) and the absolute displacement within that sector is returned in ICBAL. Note that

before performing the next I/O type of operation, those IOCB parameters should be re-established as memory address and length values.

<u>POINT</u> - sets the current file position to the exact disk address specified; the next byte read or written will be from or to that location. The absolute disk sector number is expected in the IOCB in ICBLL (LSB) and ICBLH (MSB) and the absolute displacement within that sector is expected in ICBAL. Note that before performing the next I/O type of operation, those IOCB parameters should be re-established as memory address and length values.

DISK FILENAME FORMAT

The filename portion of the device specification for the disk file-manager is of the form.

pppppppp.xxxx , where:

pppppppp = primary file name. The first character must be aplha. The following 0 to 7 characters may be any alphanumeric characters. If the primary name is less than eight characters it will padded to eight characters with blanks.

xxxx = file name extension used to classify the file type. If no extension name is given, the extension will default to blanks. If the extension name is given, then the first character must be an alpha.
The following 0-3 characters must be alphanumeric.

File Name Searching

The filename supplied by the user to FMS may contain certain "wild CARD" characters. These wild card characters allow the file name to be abbreivated. The ? character anywhere in the filename allows any character to be a match. For Example A?C is equivalent to ABC or AXC or A1C etc.. The * character causes the remainder of the primary or extension field to be padded by the ? characters. For Example, AB* would be transfored to AB??????. The filename AB, ABC, ABCDE, ABCDEFGH would all be found if AB* is designated. *.* represent any filename.

DISK FILENAME WILDCARDING

The following disk commands act upon only the first file (if any) that matches the specification:

OPEN (input, output or input/output)

GET STATUS

The following disk commands act upon all files (if any) that match the specification:

DIRECTORY LIST

DELETE

RENAME

LOCK

UNLOCK

5.7 Floppy Disk Device

This device is not accessed through CIO except through the DISK File Manager; see the USER PROGRAM/ DISK HANDLER INTERFACE MANUAL for information relating to reading & writing sectors from the floppy disk in a non-file-oriented manner.

6. Direct Device Handler Access

In some cases, it may be desired to access a device handler directly, bypassing the CIO interface. This is a straightforward thing to do, for the system resident handlers have vectors in ROM for all of their entry points. For each handler there is a set of vectors ordered as shown below:

LO Address	OPEN vector	All vectors contain the true address minus 1.
	CLOSE vector	
	GET BYTE vector	
	PUT BYTE vector	
	GET STATUS vector	
	SPECIAL vector	
HI Address	JMP INIT	

Because the vectors contain the true address minus one, a technique similar to the following one is required to access the desired routines.

```
        LDX     # vector offset     Index to desired routine
        JSR     GOHAND


GOHAND  TAY                         Save data in Register A.
        LDA     Base addr + 1, X    Push routine address to stack
        PHA
        LDA     Base addr, X
        PHA
        TYA
        RTS                         Effects a JMP
```

The base addresses for resident system handlers are shown below:

Screen Editor	-	E400	58368
Display	-	E410	
Keyboard	-	E420	
Printer	-	E430	
Cassette	-	E440	

USER PROGRAM/Disk Handler Interface

1. Introduction

2. Disk Handler Command Invocation Process — DCB

3. Disk Handler Commands

1. INTRODUCTION

The user program will use a non-CIO calling sequence to access the disk handler. This special call will pass parameters in the system Device Control Block (DCB) rather than in an IOCB.

2. DISK HANDLER COMMAND INVOCATION PROCESS

The system's single DCB is formatted as shown below (note the resemblance to IOCB format).

DCB

SERIAL * BUS I.D.	
DRIVE # 1-n	
COMMAND	
STATUS *	
BUFFER	LO
ADDRESS	HI
DISK * TIME OUT	
UNUSED	
BYTE *	LO
COUNT	HI
SECTOR	LO
NUMBER	HI

The user will insert the non-starred items into the DCB and JSR $E453.

Upon return, the status byte will indicate the degree of success of the requested operation.

The DCB parameters are described in the fillowing paragraphs.

*Need not be supplied by caller - will be filled in by disk handler.

2.1 SERIAL BUS I.D.

This number is inserted by the handler and contains the bus address of disk drive #1. It is of use only to the Serial Bus I/O Routine (SIO).

2.2 DRIVE NUMBER

This indicates which disk drive to access and is a number ranging from 1 to 4.

2.3 COMMAND BYTE

This indicates to the disk handler which command to perform (see section 3.).

2.4 BUFFER ADDRESS

This two byte pointer indicates the source or destination of disk data and status information. For disk status commands, the user need not supply an address. The disk handler will insert the address of the global status buffer.

2.5 DISK TIMEOUT

This timeout value (in whole seconds) is supplied by the disk handler for use by the SIO routine.

2.6 BYTE COUNT

This count indicates the number of bytes transferred into or out of the buffer as a result of the most recent operation.

2.7 STATUS

This system level I/O status is set by ~~I/O~~ SIO as a result of the most recent operation.

2.8 SECTOR NUMBER

This two byte number indicates to the disk handler which single sector to read or write (for read and write commands only).

3. DISK HANDLER COMMANDS

This section describes the disk commands available to the caller:

GET SECTOR

PUT SECTOR

PUT SECTOR - WITH READ CHECK

STATUS REQUEST

FORMAT DISK

3.1 GET SECTOR (Command Byte = 52)

The disk handler gets one sector's worth of data and puts it into the supplied buffer.

Status will be as shown in Appendix F.

3.2 PUT SECTOR (Command Byte = 50) Same as 3.3 except no read check.

3.3 PUT SECTOR - WITH READ CHECK (Command Byte = 57)

The disk handler puts one sector's worth of data from the supplied buffer to the disk. The disk will do a read after write check.

Status will be as shown in Appendix F.

3.4 STATUS REQUEST (Command Byte = 53)

The disk handler gets a four byte status from the disk controller and puts it in system location 'DVSTAT'.

Operation status will be as shown in Appendix F.

The disk controller status format is shown in appendix H.

3.5 FORMAT DISK (Command Byte = 21)

The disk handler commands the disk controller to format the entire disk and then to verify it. All bad sector numbers, up to a maximum count of 63, are returned and put in the supplied buffer--followed by two bytes of all ones ($FFFF_{16}$). The Byte Count parameter in the DCB will indicate the number of bytes of sector information in the buffer.(Equals number of bytes received; always 128)

Operation status will be as shown in Appendix F.

CIO/HANDLER INTERFACE GUIDE

1. <u>Introduction</u>

This document describes the interface between the Central Input/Output Monitor (CIO) and the individual device handlers and file managers that communicate with CIO. Almost all I/O operations are initiated via CIO, as that provides the most general view (largely device independent) of the I/O subsystem. CIO is entirely responsible for managing:

1. Record handling
2. User buffer maintenance
3. Logical to physical device mapping

The device handlers and file managers are then responsible for managing:

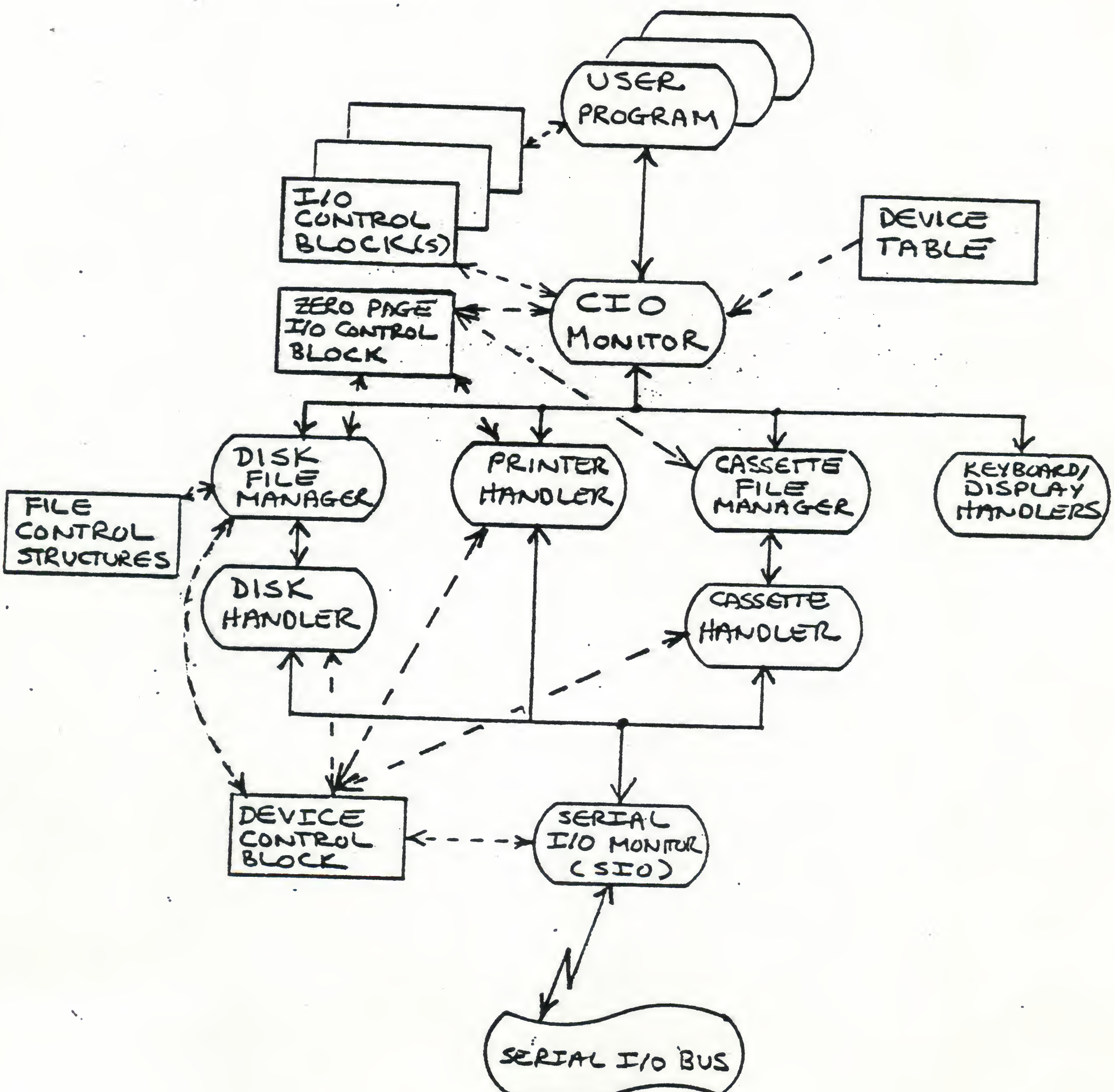
1. device buffering
2. input and/or output of a serial byte stream
3. conversion of device status to system level I/O status
4. device specific aspects of OPEN & CLOSE commands
5. device specific commands

2. <u>Central Input/Output Subsystem Structure</u>

The I/O subsystem is organized as shown below,

where  represent structural elements and

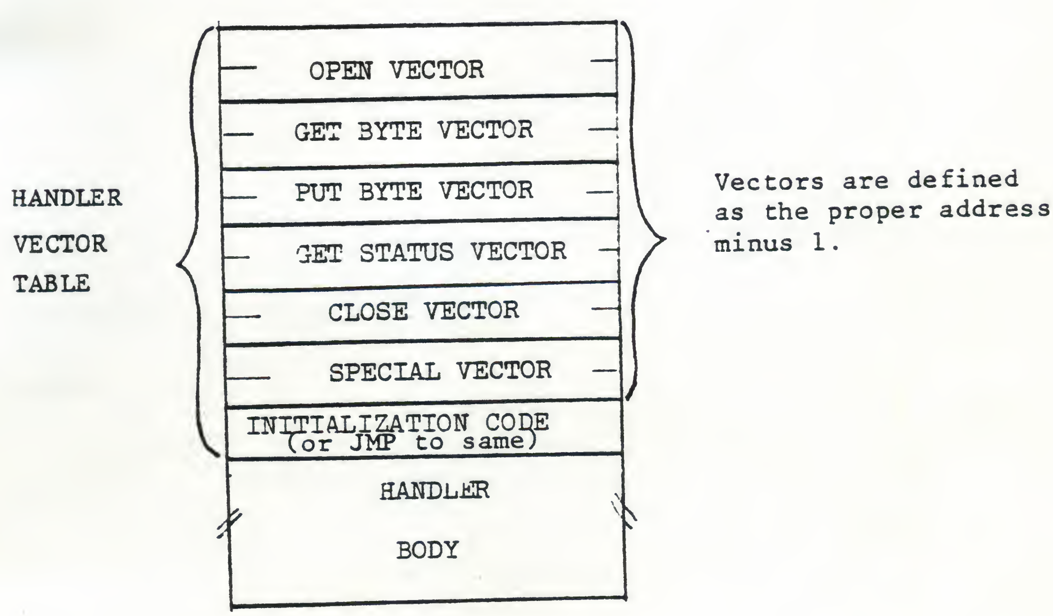
 represent processes



The user program initiates an I/O operation using one of the system I/O Control Block (IOCBs); one is required for each device or file to be concurrently accessed. The CIO monitor examines the command byte in the IOCB and resolves it to one of six handler level commands:

1. OPEN
2. GET NEXT BYTE
3. PUT NEXT BYTE
4. GET STATUS
5. CLOSE
6. SPECIAL (device specific)

Each handler has seven entry points which are organized as a table of vectors, as shown below:



The function of each of the seven vectored entry points will be described in detail in Section 3, but for now let's say that the handler acts upon the indicated command (examining the IOCB where necessary) and returns a system I/O status when the operation is complete. Communications to lower level handlers require additional structures such as the Device Control Block (DCB) or File Control Blocks (FCB). These details are covered in another document called the SIO/HANDLER INTERFACE GUIDE.

3. Handler Entry Points (Commands)

There are seven entry points for each I/O handler, each entry point corresponding to one or more command activities; the CIO/handler interface for each entry point will be described in the following paragraphs.

Data will be passed between CIO and the handler using the machine registers and a single I/O Control Block in page Ø (called ZIOCB), see Appendix A for the IOCB format.

Register A is used to pass device data, register Y is used to pass status information and register X contains the index to the originating IOCB. Upon any I/O call, CIO moves the indicated IOCB, in its entirety, to the zero page IOCB before calling the device handler; before returning to the caller, CIO moves the ZIOCB back to the originating IOCB.* Most handlers will have no need to access the originating IOCB, but will use the zero page copy. Also, handlers need not have the IOCB index in register X upon return to CIO.

* In the course of processing a given command CIO may alter the buffer address and buffer length paramaters in the ZIOCB; so, if the original values of these are required, they must be obtained from the external IOCB.

3.1 Initialization

This entry will be called after initial power-up and after a system RESET: the handler should perform whatever initialization is required of its hardware and of it's RAM data to insure proper processing of all CIO commands that follow.

3.2 <u>OPEN</u>

At handler entry, the IOCB parameters of interest will be:

Device Number	-	Contains 1 - n (for multi-device handlers)
Command Byte	-	OPEN command
Buffer Address	-	Points to file name specification (See Appendix D)
Auxilliary Inf.	-	Indicates OPEN direction Contains device specific information

The handler will attempt to perform the indicated OPEN and will signal the success of the operation by returning an IOCB status in register Y (See Appendix C).

The responsibility for checking for multiple OPENs to the same device or file, where it is illegal, lies with the handler or file manager, not CIO.

File managers will return file status in the Auxilliary Information Bytes (see Appendix I).

3.3 GET Next Byte

At handler entry, the IOCB parameter of interest will be:

Device Number - contains 1 - n (for multi-device handlers)

The handler will get a byte of data from it's device buffer, if one is available, or else initiate a read to the device to get more data.

Handlers that do not have timeouts associated with reading data, such as Keyboard and Cassette, must monitor the system BREAK flag and set the appropriate status and return when a BREAK condition occurs before data is available.

When the data byte is available, the handler will return to CIO with the data in register A and the status in register Y.

CIO checks for reads from devices or files that have not been OPENed, or OPENed for output only; the handler will not be called in those cases.

3.4 Put Next Byte

At handler entry, the IOCB parameter of interest will be:

Device Number - contains 1 - n (for multi-device handlers)

The handler will put the byte of data in register A into it's device buffer or send the data to the device. When device buffers fill, they are sent to the device before returning to CIO.

The handler will return to CIO with the status in register Y.

CIO checks for writes to devices or files that have not been OPENed, or OPENed for input only; the handler will not be called in those cases.

3.5 Get Status

At handler entry, the IOCB parameters of interest will be:

Device number	-	contains 1 - n (for multi-device handlers)
Buffer address	-	points to filename specification if device not already OPEN.

The handler or file manager will get device status from the device controller and put the status bytes (maximum of four) in system locations "DVSTAT" through "DVSTAT + 3".

The handler will return to CIO with the operation status in register Y.

Note that a device or file need not be OPEN to perform a GET STATUS operation.

3.6 CLOSE

At handler entry, the IOCB parameter of interest will be:

Device number	-	contains 1 - n (for multi-device handlers)

The handler will release any held resources that relate specifically to that device or file, and for OPEN outputs: 1) send any remaining data in local device buffers to the device and 2) mark the end-of-file. The handler will also mark the device as not OPEN and take any other device specific action normally associated with CLOSE. CIO will mark the IOCB as not open.

3.7 Special

This handler entry is used to support all functions not handled by the other entry points, such as file RENAME, file DELETE, etc. The handler will interrogate the IOCB Command Byte for legality and perform the indicated command, if possible.

The handler will return to CIO with the operation status in register Y.

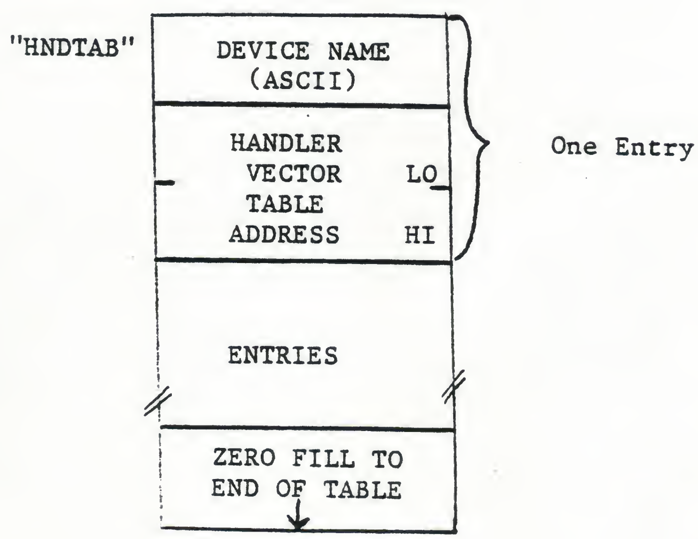
Devices need not be OPEN nor CLOSED to perform special operations using CIO; the handler must check, where there are restrictions.

4. Non-Resident Handlers

Non-resident handlers reside in cartridges, disk or RAM and are known to the CIO subsystem only if they have an entry in the System Handler Table.

This table contains the names and addresses for each of the devices supported by the system. The table is configured at power-up time to contain entries for the ROM resident system handlers.

The table format is shown below:



Power-up adds entries to the table starting with the beginning (low address); CIO scans the table from the end to the beginning. Thus, in the case of multiple handler entries for a given device, the entry nearest the end of the table will be selected.

The table has room for ten entries, five of which are filled at power-up time (with K:, E:, S:, C:, and P:); the disk handler will optionally take one more entry (D:).

To install a new handler, some procedure must place the symbolic device name and the handler vector table address into the System Handler Table. This is the responsibility of the program supplying the handler. The installation can occur at cartridge initialization (for cartridge resident handlers), but may be deferred until later, if desired.

Handler/SIO Interface Guide

1. Introduction

2. Central Input/Output Subsystem Structure

3. Handler/SIO Interface

4. SIO Operation

1. Introduction

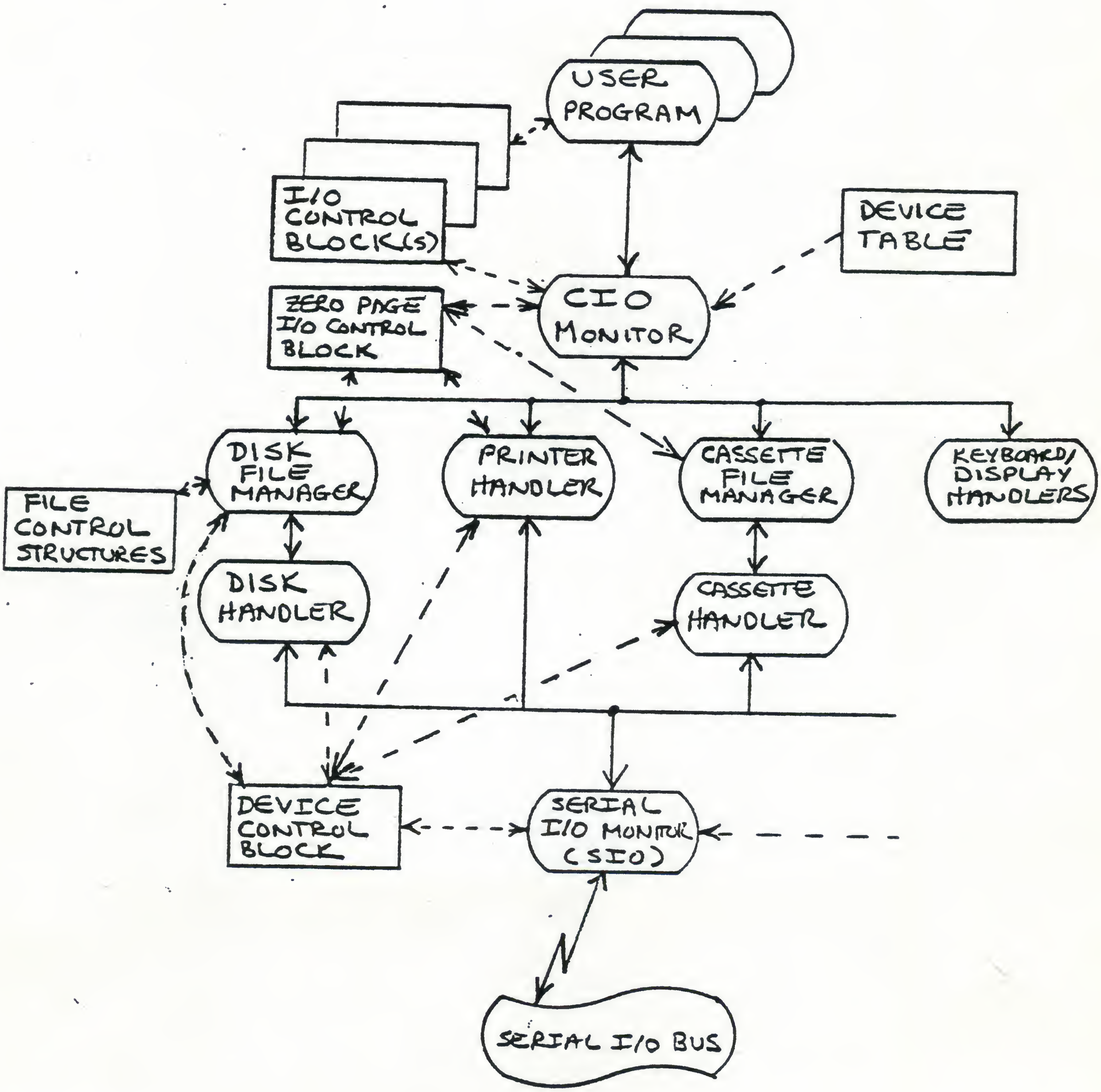
This document describes the interface between the serial bus device handlers and the serial I/O Bus Monitor (SIO). All bus transactions are initiated and terminated by SIO following a bus protocal that is device independent; this protocol is defined in the SIO BUS PROTOCOL DESCRIPTION document.

SIO is responsible for handling:

1. Serial bus protocol and chip management
2. Error retries
3. Time-out functions
4. Data transfer between handler buffers and the serial bus.

2. <u>Central Input/Output Subsystem Structure</u>

The I/O subsystem is organized as shown below, where ☐ represent structural elements and ▢ represent processes.



The handler initiates an I/O operation using the system Device Control Block (DCB) to pass parameters to and from SIO. The parameters are described in the next section.

3. <u>Handler/SIO Interface</u>

The Device Control Block is formatted as shown below:

SERIAL BUS I.D.	
UNIT # (1-n)	
SERIAL BUS COMMAND	
STATUS	
HANDLER BUFFER ADDRESS	LO HI
DEVICE TIMEOUT	
NOT USED	
BYTE COUNT	LO HI
AUXILLIARY DATA	

The DCB parameters are described in the following paragraphs.

3.1 Serial Bus I.D.

This must contain the Serial Bus address of unit #1 of the device to be accessed.

3.2 Unit Number

This indicates which of n units of a device type to access. SIO will access the device number calculated by adding the Serial Bus I.D. to the Unit Number and subtracting one.

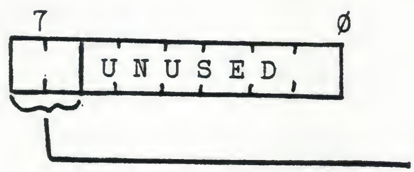
3.3 Serial Bus Command

This byte will be sent to the device controller as part of the serial bus command frame.

3.4 Status

The status byte is bi-directional; the handler will use it to indicate to SIO what to do after the command frame is sent and SIO will use this byte to indicate the status of the operation.

Prior to SIO call:



ØØ = no data transfer associated with operation.

Ø1 = data expected from device

1Ø = device will expect data

11 = invalid

After SIO call:

See appendix G for status codes.

3.5 <u>Buffer Address</u>

This two byte pointer indicates the source or destination of device data and status information.

3.6 <u>Device Timeout</u>

This timeout value (in whole seconds) defines the maximum allowable time to wait for any single command attempt.

3.7 <u>Byte Count</u>

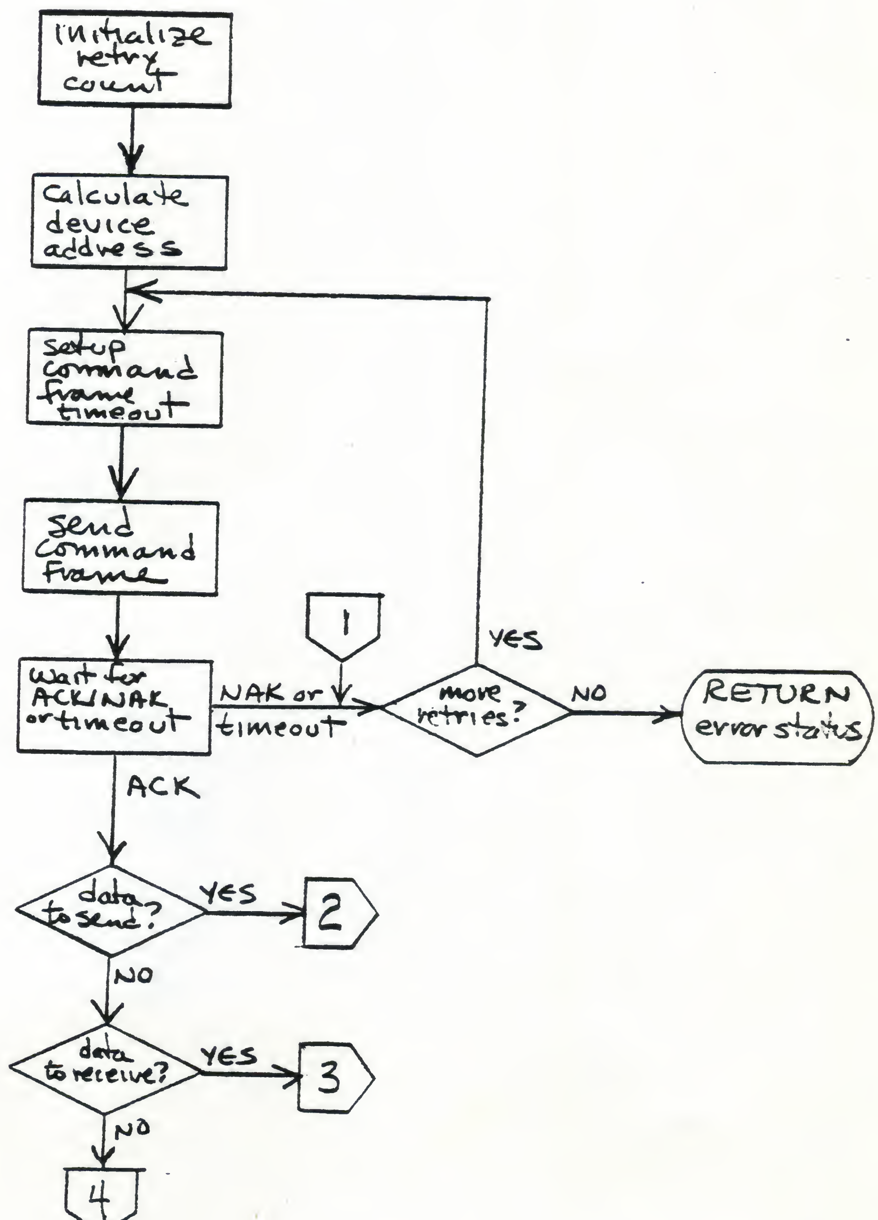
This count indicates the number of bytes to transfer into or out of the buffer for the current operation. Not used if STATUS byte indicates that no data transfer is to take place.

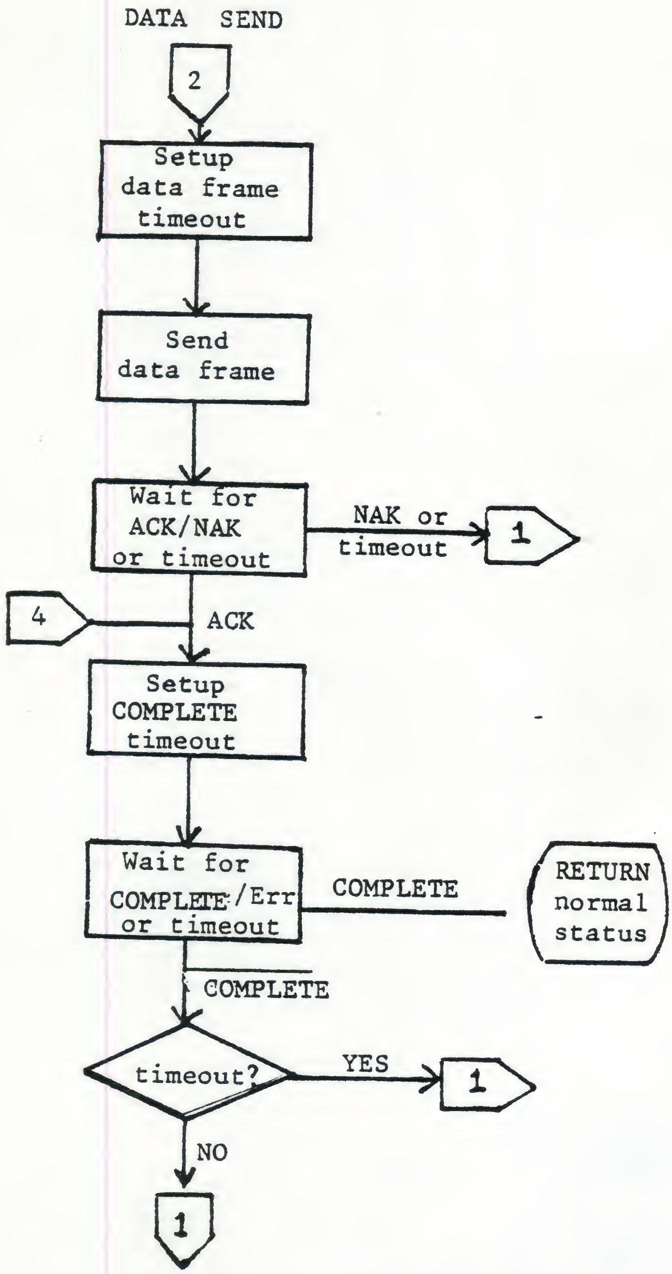
3.8 <u>Auxilliary Data</u>

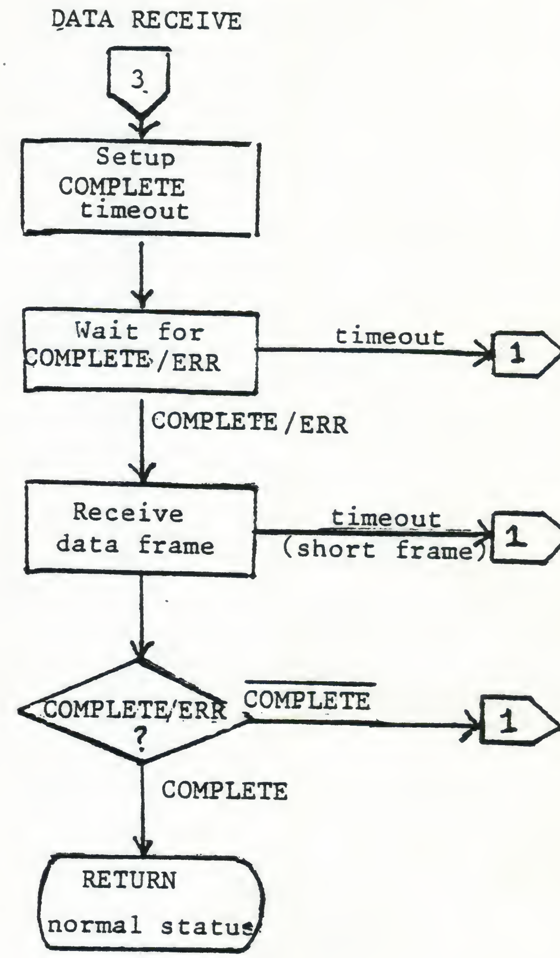
These two bytes are included in the Serial Bus command Frame by SIO; they have device specific meanings.

4. SIO Operation

SIO's operation is represented by the flowchart below; this applies to all devices except for the modem and cassette, which are special cased.




DATA SEND
2
Setup data frame timeout
Send data frame
Wait for ACK/NAK or timeout
NAK or timeout
1
4
ACK
Setup COMPLETE timeout
Wait for COMPLETE/Err or timeout
COMPLETE
RETURN normal status
COMPLETE
timeout?
YES
1
NO
1


DATA RECEIVE
3
Setup COMPLETE timeout
Wait for COMPLETE/ERR
timeout
1
COMPLETE/ERR
Receive data frame
timeout (short frame)
1
COMPLETE/ERR ?
COMPLETE
1
COMPLETE
RETURN normal status

SERIAL I/O BUS PROTOCOL

1. Introduction

2. Bus Commands

3. Bus Timing

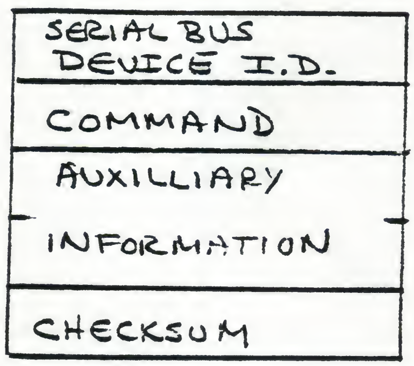
4. Current Devices

2. Bus Command Types

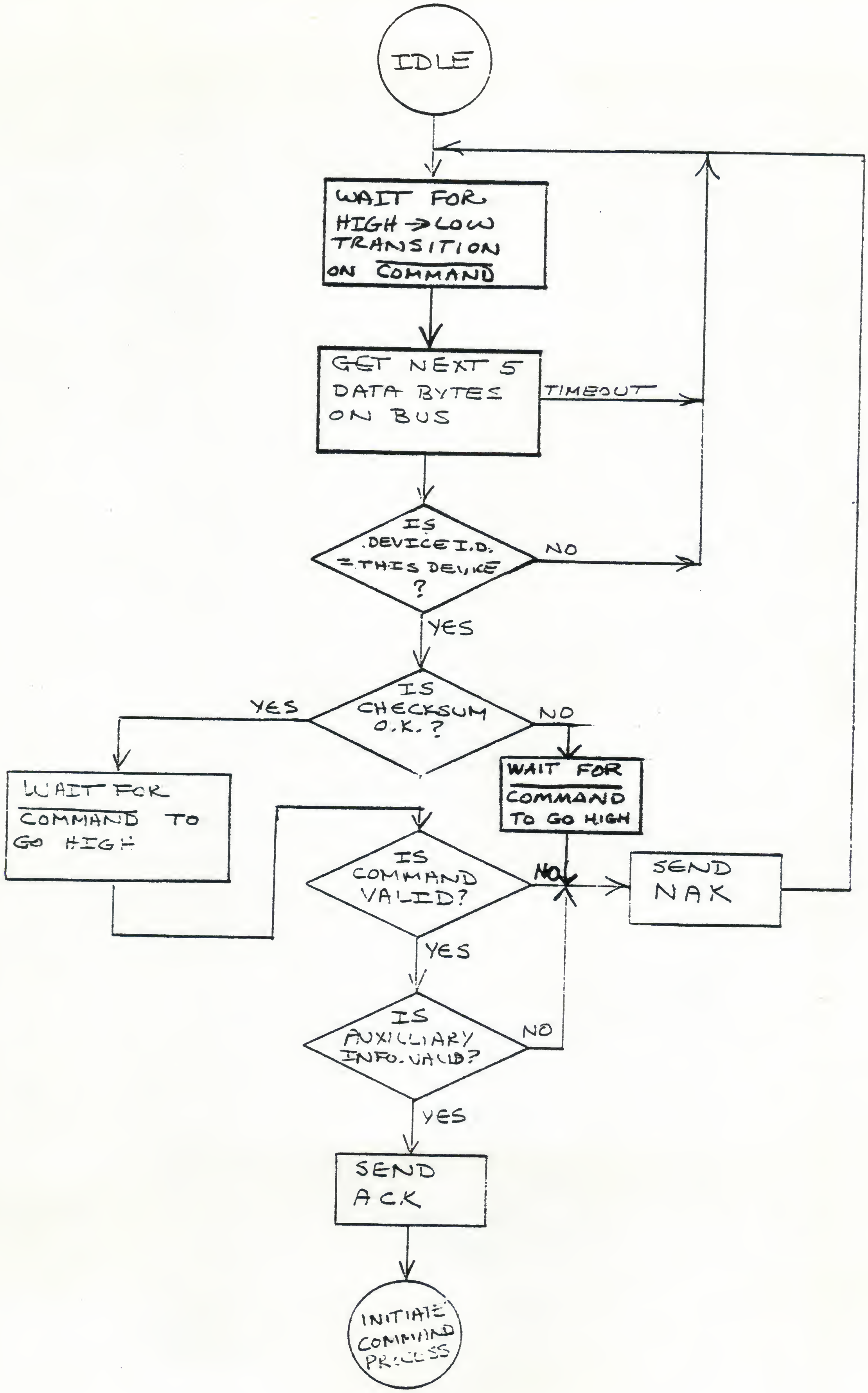
The Serial Bus Protocol accomodates three basic types of commands: 1) data send, 2) data receive, and 3) immediate (no data - command only).

The common element in all three command types is the command frame, five bytes of information sent while the $\overline{\text{COMMAND}}$ line is held low.

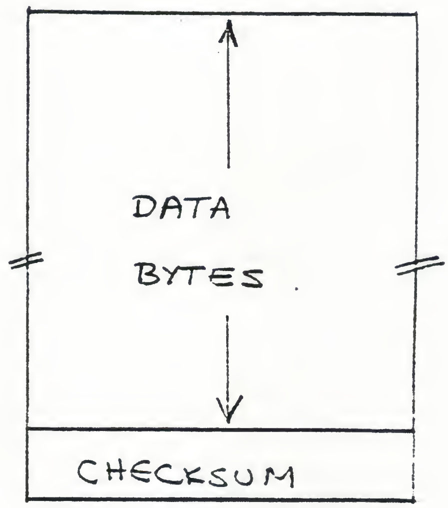
The command frame format is shown below:



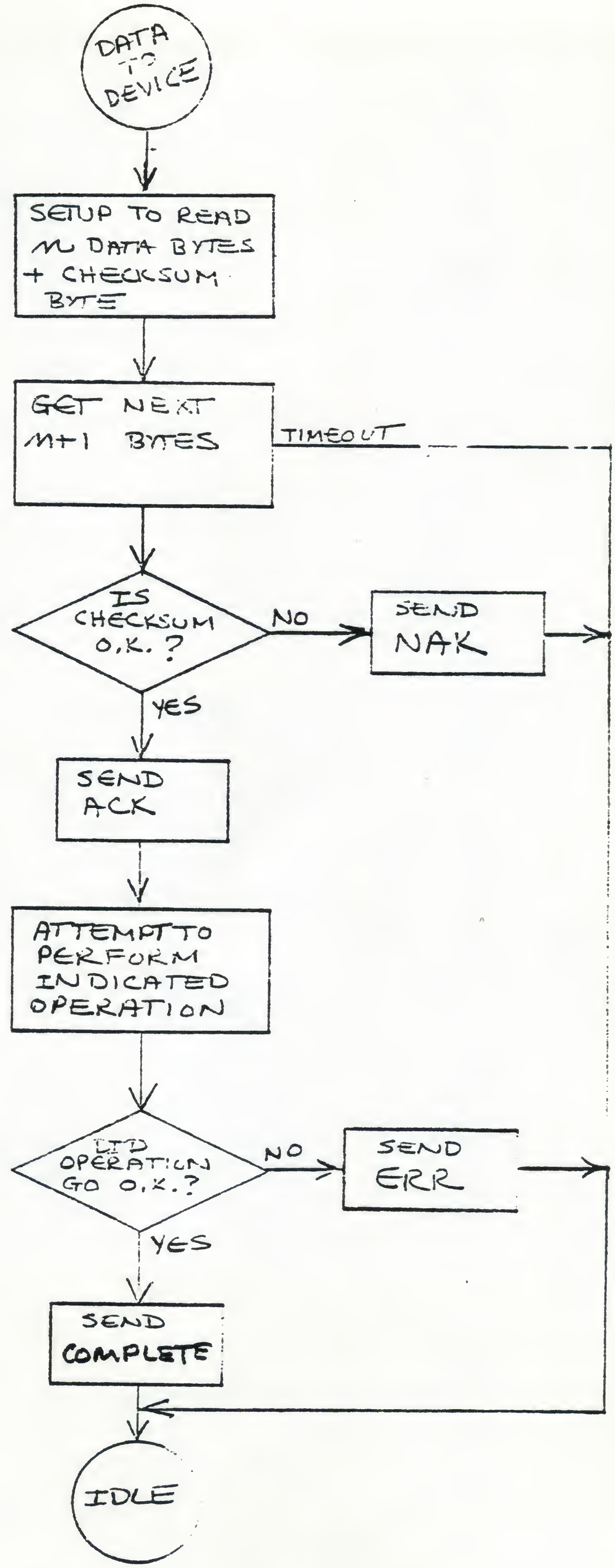
The flowchart on the next page shows a typical device controller's processing of a command frame.


IDLE
WAIT FOR HIGH → LOW TRANSITION ON COMMAND
GET NEXT 5 DATA BYTES ON BUS
TIMEOUT
IS DEVICE I.D. = THIS DEVICE ?
NO
YES
IS CHECKSUM O.K. ?
YES
NO
WAIT FOR COMMAND TO GO HIGH
WAIT FOR COMMAND TO GO HIGH
IS COMMAND VALID?
NO
SEND NAK
YES
IS AUXILLIARY INFO. VALID?
NO
YES
SEND ACK
INITIATE COMMAND PROCESS

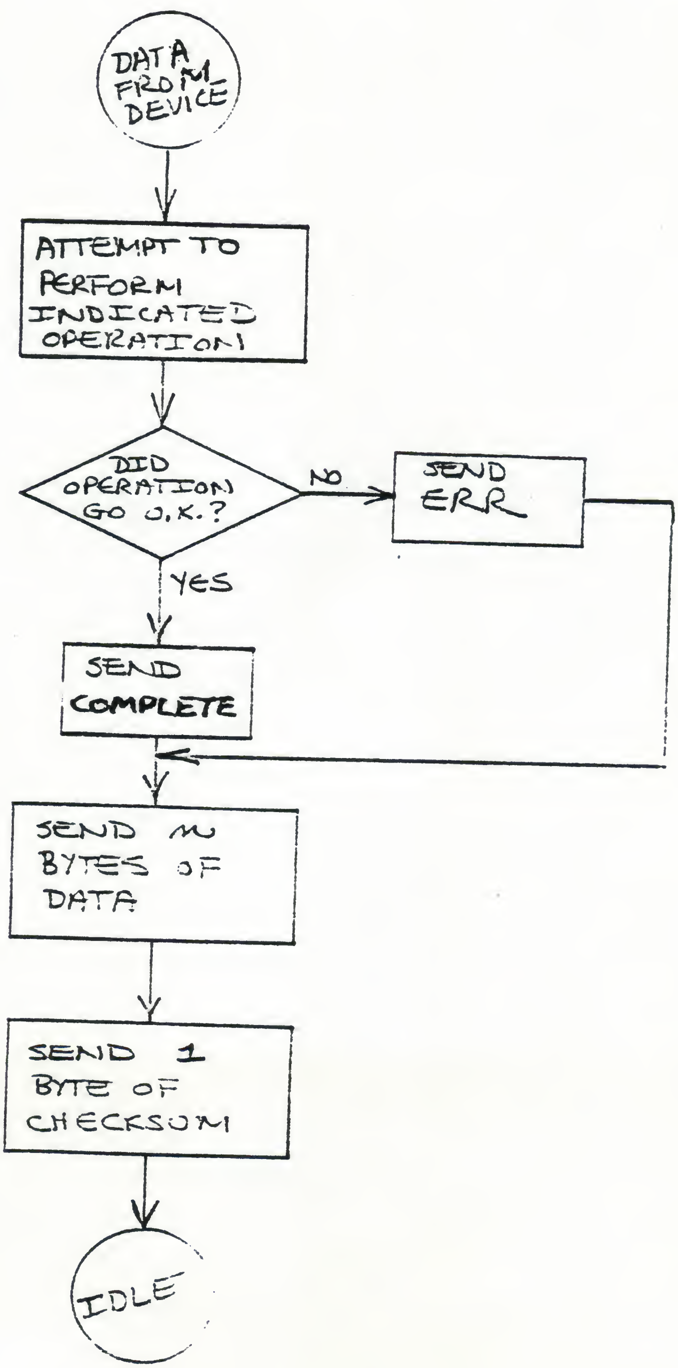
Following the command frame is an optional data frame which is formatted as shown below:



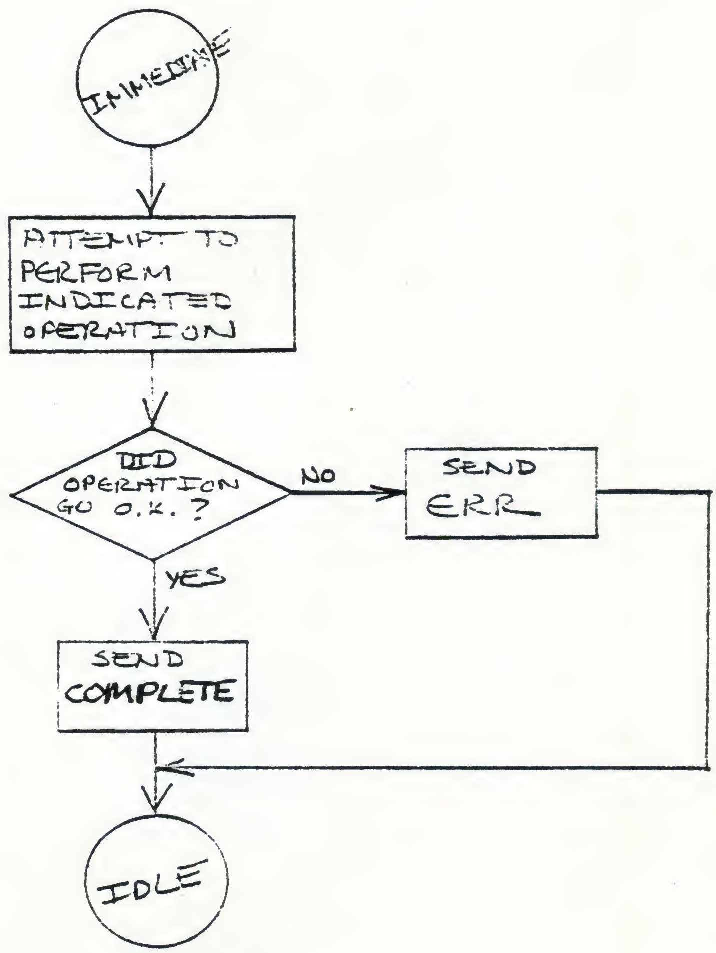
This data frame may originate at the central processor or at the device, depending upon the command. The flowchart on the next page shows a typical device controller's processing of a data frame from the central processor.


DATA TO DEVICE
SETUP TO READ n DATA BYTES + CHECKSUM BYTE
GET NEXT n+1 BYTES
TIMEOUT
IS CHECKSUM O.K.?
NO
SEND NAK
YES
SEND ACK
ATTEMPT TO PERFORM INDICATED OPERATION
DID OPERATION GO O.K.?
NO
SEND ERR
YES
SEND COMPLETE
IDLE

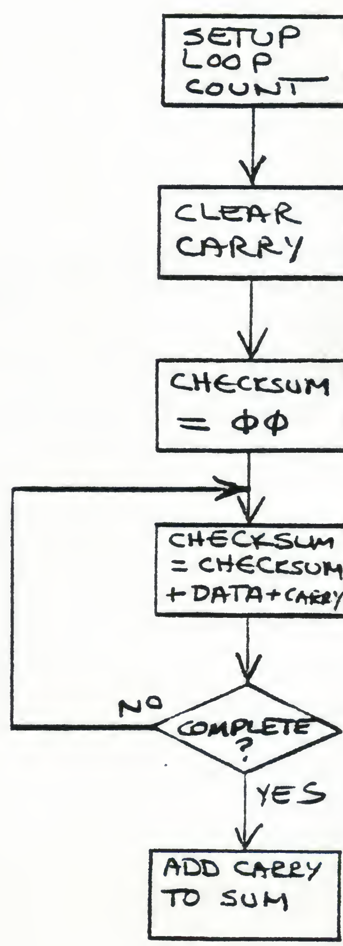
The flowchart at the bottom of this page shows a typical device controller's processing of a data frame to be sent to the central processor.



The flowchart at the bottom of this page shows a typical device controller's of an "immediate" type of command - no data frame.



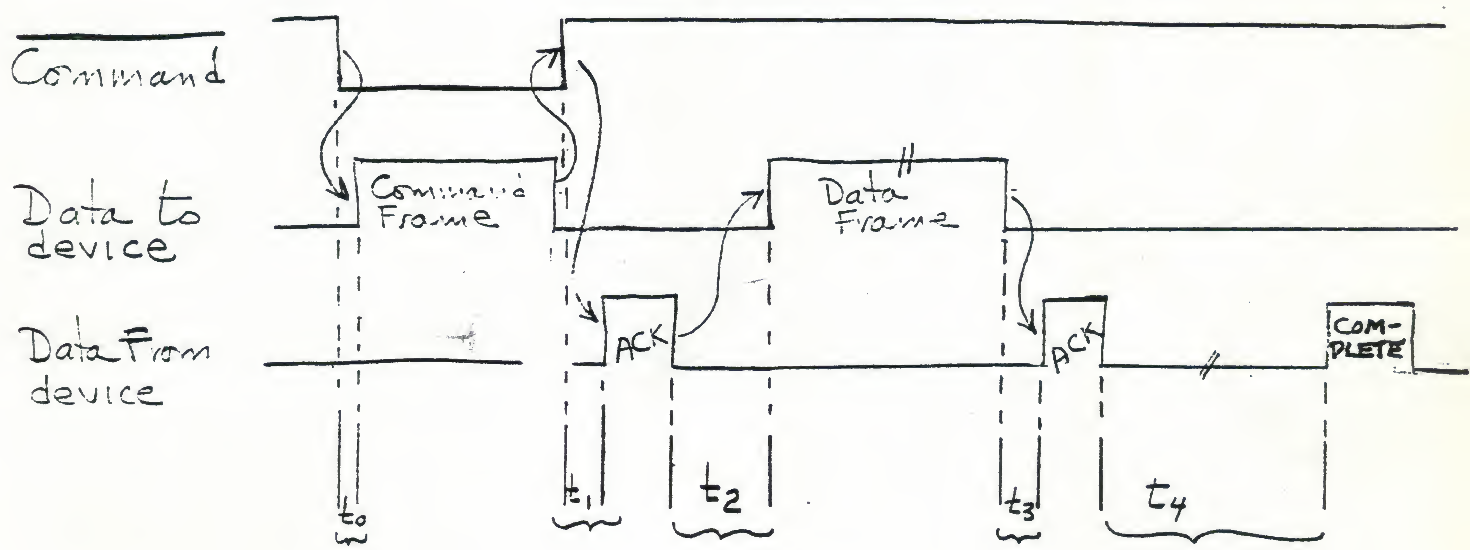
All checksums used as part of the serial bus protocol are simple 8-bit arithmetic sums using the carry bit, as shown below:



3. Bus Timing

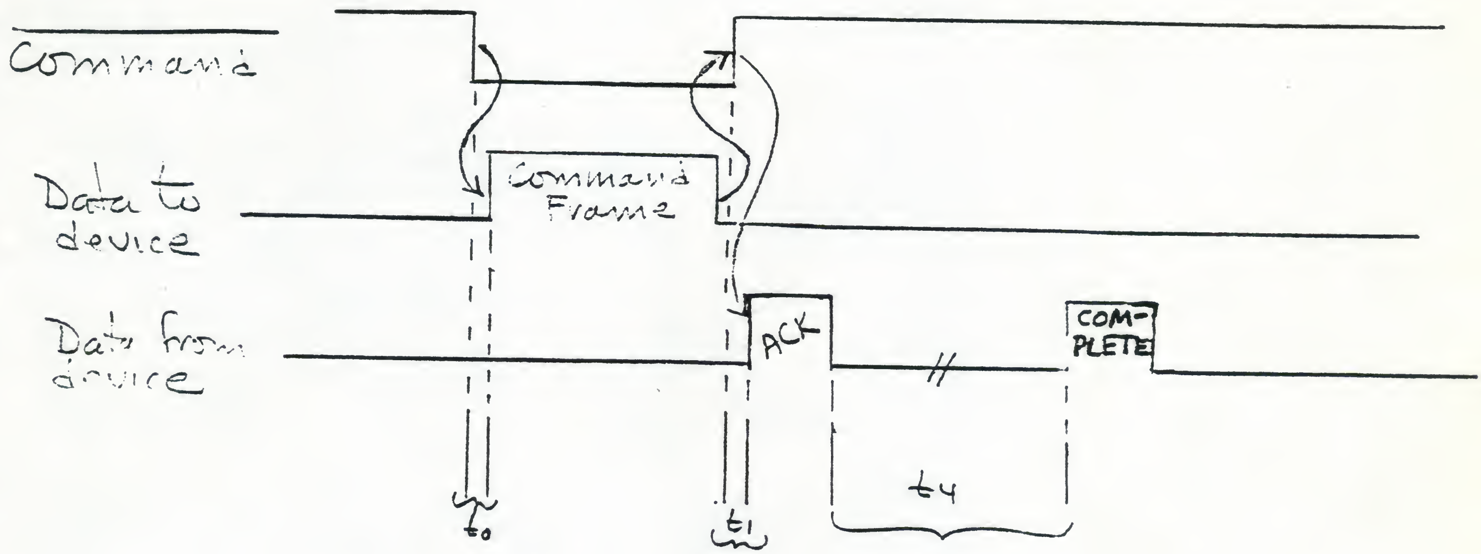
The serial Bus Protocol accomodates three basic types of commands: 1) data send, 2) data receive, and 3) immediate (no data frame). A timing diagram is given below for each type:

Data Send

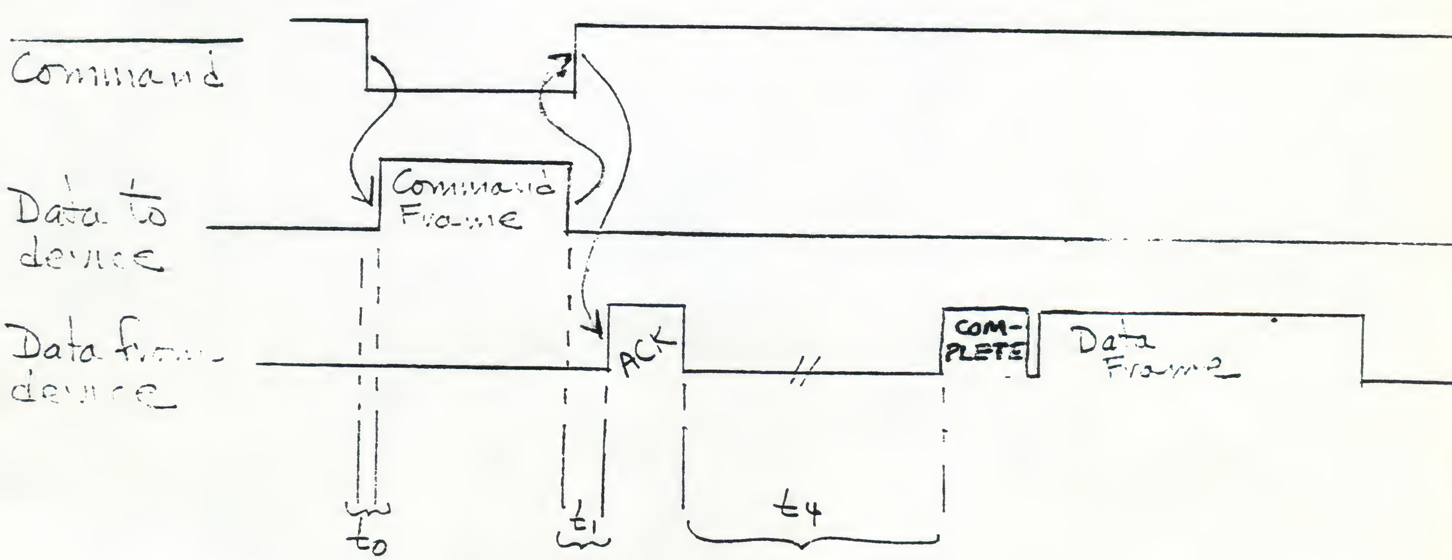


to (command frame initiate delay)	= 100 usec - 500 usec
t_1 (command frame acknowledge delay)	= Ø - 500 usec
t_2 (controller data receive setup time)	= 25Ø u - 1 sec.
t_3 (data frame acknowledge delay)	= 150 u - 15 msec.
t_4 (operation complete)	= 150 u - 00

Immediate



Data Receive



4. Current Devices

This section provides information on the two current intelligent serial bus devices, the 40 column printer and the floppy disk.

4.1 Printer (I.D. = $4\emptyset_{16}$)

The printer controller recognizes two commands: WRITE and STATUS REQUEST.

4.1.1 The printer WRITE command frame contains the following values:

Command = 57_{16}

Auxilliary Byte 1 = print option

Auxilliary Byte 2 = undefined (not used)

Auxilliary Byte 1 (AUX1) will contain one of the following values -

$4E_{16}$ = Normal print (40 chars/line)

53_{16} = Sideways print(16 chars/line)

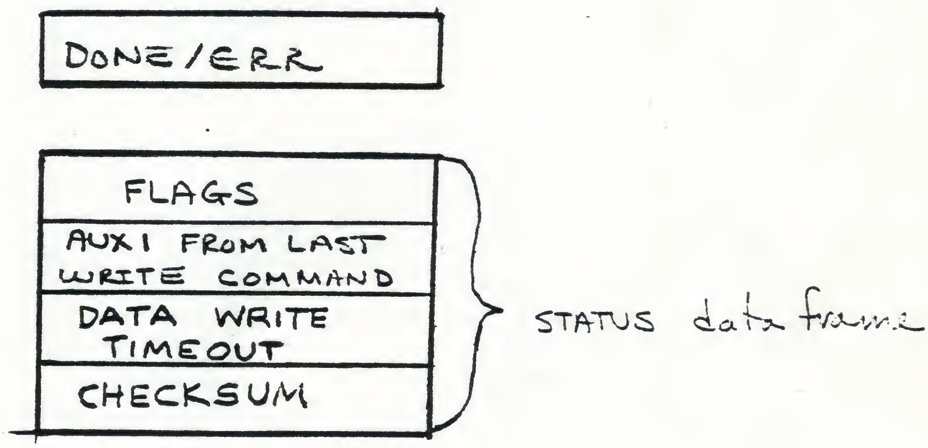
The printer WRITE data frame contains either 29 data bytes plus a checksum byte, or 40 data bytes plus a checksum byte, depending upon the value of AUX1 in the command frame.

4.1.2 The printer STATUS REQUEST command frame contains the following values:

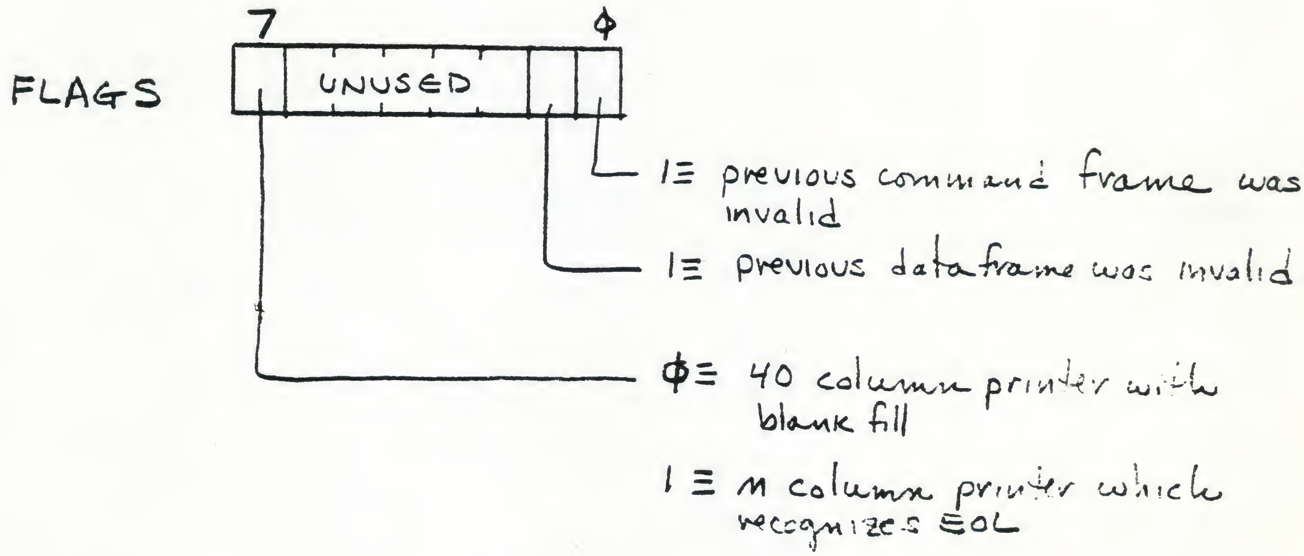
Command = 53_{16}

Auxilliary Bytes = undefined (not used)

The printer controller will return the status as a valid data frame as shown below.



The FLAGs byte contains information relating to the most recent command prior to the STATUS REQUEST and some controller constants. The FLAG byte is formatted as shown.



Data Write Timeout = maximum time to print a line of data assuming worst case controller produced cool-off delay. (1 sec.)

4.2 Floppy Disk Controller (I.D.s = $31\text{-}34_{16}$)

The floppy disk controller recognizes commands: READ, PUT, WRITE, FORMAT, and STATUS REQUEST.

4.2.1 The disk READ command frame contains the following values:

Command = 52_{16}

$\left.\begin{array}{l}\text{Auxilliary Byte 1 = Disk Sector L.S.B}\\ \text{Auxilliary Byte 2 = Disk Sector M.S.B.}\end{array}\right\}$ (1 - 72Ø)

The disk READ data frame contains 128 bytes of data plus a checksum byte.

4.2.2 The disk PUT command frame contains the following values:

Command = 50_{16}

$\left.\begin{array}{l}\text{Auxilliary Byte 1 = Disk Sector L.S.B.}\\ \text{Auxilliary Byte 2 = Disk Sector M.S.B.}\end{array}\right\}$ (1-720)

4.2.3 The disk WRITE command (read after write check) frame contains the following values:

Command = 57_{16}

$\left.\begin{array}{l}\text{Auxilliary Byte 1 = Disk Sector L.S.B.}\\ \text{Auxilliary Byte 2 = Disk Sector M.S.B.}\end{array}\right\}$ (1-720)

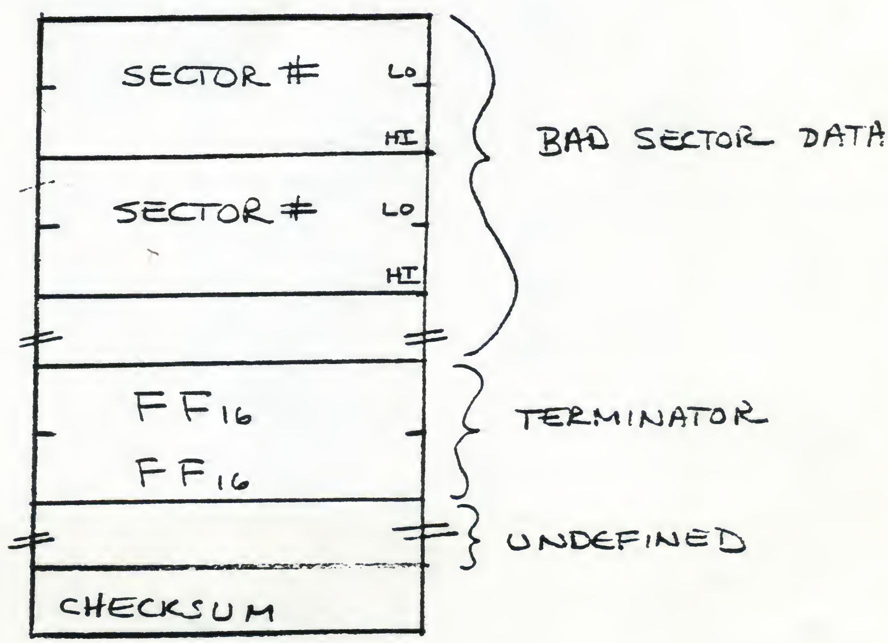
The disk WRITE data frame contains 128 bytes of data plus a checksum byte.

4.2.4 The disk FORMAT command frame contains the following values:

Command = 21_{16}

Auxilliary Bytes = undefined (unused)

The disk FORMAT data frame (returned by the controller) contains 128 bytes of data plus a checksum byte, formatted as shown below:



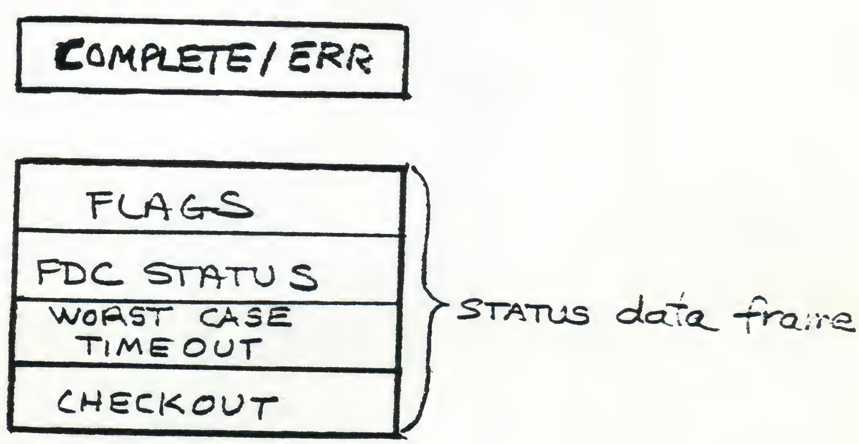
The disk is formatted and then verified by the disk controller; the FORMAT data frame contains from zero to 63 bad-sector numbers, followed by two bytes of all ones, followed by fill bytes to pad out the data block to 128 bytes, followed by a checksum. The sector numbers may be in any order.

4.2.5 The disk STATUS REQUEST command frame contains the following values:

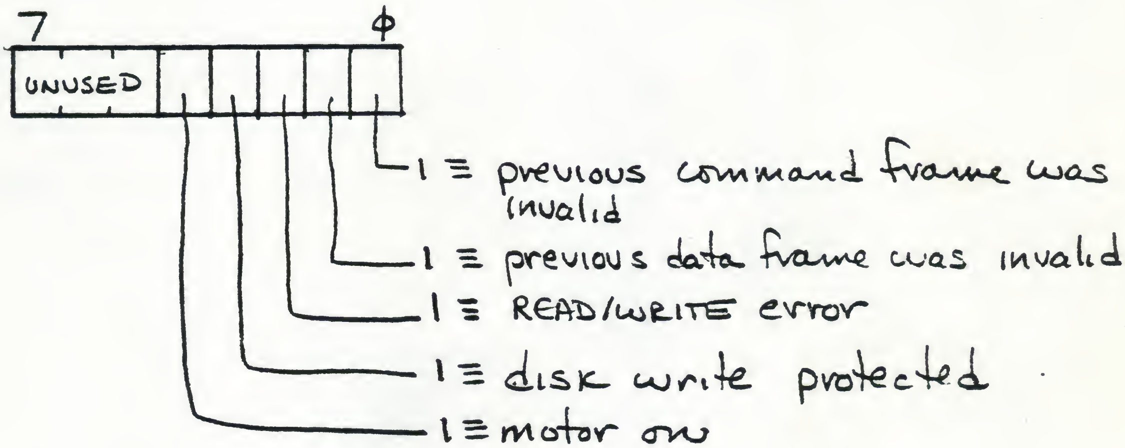
Command = 53_{16}

Auxilliary Bytes = undefined (unused)

The disk controller will return the status as a valid data frame as shown below.



The FLAGS byte contains information relating to the most command prior to the STATUS REQUEST, and is formatted as shown.



Worst case timeout	=	maximum time to wait for slowest operation (FORMAT). (1 sec.)

Appendix A - IOCB Format

	Field			Initiating IOCB Name	Zero-Page IOCB Name
	HANDLER I.D.	*		ICHID	ICHIDZ
	DEVICE #	*		ICDNO	ICDNOZ
	COMMAND			ICCOM	ICCOMZ
	STATUS	*		ICSTA	ICSTAZ
128	BUFFER	LO		ICBAL	ICBALZ
5	ADDRESS	HI		ICBAH	ICBAHZ
163	PUT BYTE	LO*	A3	ICPTL	ICPTLZ
246	ADDRESS	HI*	F6	ICPTH	ICPTHZ
	BUFFER	LO		ICBLL	ICBLLZ
	LENGTH	HI		ICBLH	ICBLHZ
	AUXILLIARY	LO		ICAX1	ICAX1Z
	INFORMATION	HI		ICAX2	ICAX2Z

*Supplied by CIO, not calling program

$Ø34Ø	IOCB	Ø	Always initialized for Editor
Ø35Ø	"	1	
Ø36Ø	"	2	
Ø37Ø	"	3	
Ø38Ø	"	4	
Ø39Ø	"	5	
Ø3AØ	"	6	
Ø3BØ	"	7	

Appendix B - IOCB Command Bytes

Ø3	-	OPEN
Ø5	-	GET RECORD
Ø7	-	GET CHARACTER(S)
Ø9	-	PUT RECORD
ØB	-	PUT CHARACTER(S)
ØC	-	CLOSE
ØD	-	STATUS REQUEST
2Ø	-	RENAME
21	-	DELETE
22	-	FORMAT
23	-	LOCK FILE
24	-	UNLOCK FILE
25	-	POINT
26	-	NOTE
11	-	DRAW LINE
12	-	FILL

Appendix C - IOCB Status Bytes

	01	- OPERATION COMPLETE (no errors)
	02	- TRUNCATED RECORD
128	80	- BREAK KEY ABORT
130	82	- NON-EXISTENT DEVICE
132	84	- INVALID COMMAND
133	85	- DEVICE OR FILE NOT OPEN
134	86	- INVALID IOCB NUMBER (not stored in IOCB!)
136	88	- END-OF-FILE
138	8A	- DEVICE TIMEOUT (Device doesn't respond)
139	8B	- DEVICE NAK
140	8C	- SERIAL BUS INPUT FRAMING ERROR
142	8E	- SERIAL BUS DATA FRAME OVERRUN ERROR
143	8F	- SERIAL BUS DATA FRAME CHECKSUM ERROR
144	90	- DEVICE DONE ERROR (Responds with invalid DONE byte)
147	93	- INSUFFICIENT MEMORY FOR SCREEN MODE OR BAD SCREEN MODE
160	A0	- DRIVE # ERROR
161	A1	- TOO MANY OPEN FILES (NO SECTOR BUFFER AVAILABLE)
162	A2	- MEDIUM FULL (NO FREE SECTORS)
163	A3	- FATAL SYSTEM DATA I/O ERROR
164	A4	- FILE # MISMATCH
165	A5	- FILE NAME ERROR
166	A6	- POINT DATA LENGTH ERROR
167	A7	- FILE LOCKED
168	A8	- COMMAND INVALID (SPECIAL OPERATION CODE)
169	A9	- DIRECTORY FULL (64 FILES)
171	AB	- POINT INVALID

Appendix D - Device/filename Specification

At OPEN time, the buffer pointer contains the address of the start of a device/filename specification. The specification consists of ASCII characters in the following format:

<device specifier> [<device number >] : [<filename>] EOL

where:

<device specifier> = letter device specifier ⟶ C ≡ Cassette
M ≡ Modem
etc.

<device number> = 1 - n (defaults to 1 if not specified)

<filename> = name of file

Appendix G - SIO Status Codes

8A - Device Timeout (Device doesn't respond)

8B - Device NAK

8C - Serial Bus Input Framing Error

8E - Serial Bus Data Frame Overrun Error

8F - Serial Bus Data Frame Checksum Error

90 - Device DONE Error (Responds with invalid DONE byte)

Appendix H - Device Status Bytes

DISK - Byte 1

invalid command frame
invalid data frame
unsuccessful write
write protect
active/standby?

Byte 2 Controller chip status

Byte 3 timeout

Byte 4

PRINTER Byte 1

invalid command frame
invalid data frame
intelligent printer controller

Byte 2 AUX1 from previous command

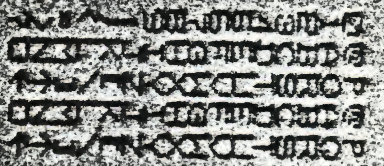
Byte 3 timeout

Byte 4

Appendix I - Printer Character Set

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~

5 OPEN #1,8,0,"P:"
10 FOR I=0 TO 128 STEP 16
20 FOR J=I TO I+15
30 PRINT #1,CHR$(J);
40 NEXT J
45 PRINT #1," "
50 NEXT I
60 END
```



```
5 OPEN #1,8,83,"P:"
10 FOR I=0 TO 128 STEP 16
20 FOR J=I TO I+15
30 PRINT #1,CHR$(J);
40 NEXT J
45 PRINT #1," "
50 NEXT I
60 END
```

Appendix J- Serial Bus Device Address

Floppy Disks	$31 - 34_{16}$
Printer	$4\emptyset_{16}$
Modem	$5\emptyset - 58_{16}$

Appendix K - Serial Bus Command Codes/Control Codes

Commands

PUT	50	('P')
READ	52	('R)
WRITE	57	('W')
STATUS	53	('S')
FORMAT	21	('!')

Control Codes

ACK	41	('A')
NAK	4E	('N')
COMPLETE	43	('C')
ERR	45	('E')

GRAPHICS FOR CONTRO[L]
CHARACTERS:

Esc	␛	→	→
↑	↑	CLEAR	↰
↓	↓	BACK S.	◀
←	←	TAB	▶

Appendix L- Atari External ASCII

	ΦX	2X	4X	6X	8X	AX	CX	EX
ΦΦ		b̸	@	◆				
Φ1		!	A	a				
Φ2		"	B	b				
Φ3		#	C	c				
Φ4		$	D	d				
Φ5		%	E	e				
Φ6		&	F	f				
Φ7		'	G	g				
Φ8		(	H	h				
Φ9		)	I	i				
ΦA		*	J	j				
ΦB		+	K	k				
ΦC		,	L	l				
ΦD		-	M	m				
ΦE		.	N	n				
ΦF		/	O	o				
1Φ		Φ	P	p				
11		1	Q	q				
12		2	R	r				
13		3	S	s				
14		4	T	t				
15		5	U	u				
16		6	V	v				
17		7	W	w				
18		8	X	x				
19		9	Y	y				
1A		:	Z	z				
1B	ESC	;	[	♣	EOL			
1C	↑	<	\	|	DEL LINE			
1D	↓	=	]	CLEAR	INS LINE			(BELL)
1E	←	>	^	BACKSP.	CLR TAB			DEL CHAR
1F	→	?	_	TAB	SET TAB			INS CHAR